КОМПЬЮТЕР ПРЕСС



Чувствуете, друзья-соотечественники, что как-то неуютно и неудобно стало работать с любимым компьютером? Чего-то не хватает... А может быть это что-то и есть интегрированная система МАСТЕР?

Интегрированная система МАСТЕР позволяет обрабатывать информацию различных типов: тексты, электронные таблицы, графики, рисунки. ИС МАСТЕР имеет мощную СУБД и средства работы в локальных вычислительных сетях. Система ориентирована на отечественного пользователя. Русскоязычный интерфейс системы реализуется без всякой аппаратной настройки. Единая визуальная среда для структурирования, обработки и хранения информации и программ позволяет работать в ИС МАСТЕР как профессиональным программистам, так и начинающим.



113184 Москва, ул. Островского, 44
Телефон: (095) 233-00-06
Факс: (095) 235-10-53
Телекс: 411660 MICRO
E-mail: lexicon@micro.msk.su

КОМПЬЮТЕР ПРЕСС

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Работа с динамической памятью в Borland C++ 3

DOSSHELL — окно в "Окна" 13

Использование динамических виртуальных методов в Turbo Vision 17

Turbo Pascal в среде Windows 19

Что такое системы автоматического чтения текстов? или Еще раз про OCR 23

ТЕНДЕНЦИИ

Что было в прошлом году на компьютерном рынке? 30

АППАРАТНОЕ ОБЕСПЕЧЕНИЕ

Микропроцессоры Intel: из прошлого в будущее 31

КОМПЬЮТЕРНЫЕ ВИРУСЫ

Компьютерные вирусы-92: от плохого к худшему, или Есть ли вирусы на Марсе? 35

ВЫСТАВКИ

Репортаж с Комдекса, или Выставка в пустыне 39

НОВЫЕ ПРОДУКТЫ

Там, где кончается OWL,— начинается OGL... 43

ВПЕЧАТЛЕНИЯ

Компьютер Encyclopedia 49

КАК ЭТО РАБОТАЕТ

Струйные принтеры 53

СЕТИ

Локальная сеть и шахматы, или Почему вы не используете NetBIOS? 57

ИГРЫ

Sierra: более 10 лет в индустрии компьютерных игр 61

ПОРТРЕТ ФИРМЫ

Computer Associates. Фирма и люди 63

ПЕРСОНАЛИИ

Computer Associates. Фейерверк программных продуктов 65

Знакомьтесь: фирма TurboPower Software 69

МУЛЬТИМЕДИА

Мультимедиа в трех измерениях 70

КОЛОНКА РЕДАКТОРА 76

НОВОСТИ 77

3'93



КОМПЬЮТЕРПРЕСС

Издается с 1989 года
Выходит 12 раз в год
3'93 (39)

Адрес редакции:
113093 Москва, аб.ящик 37
Факс: (095) 200-22-89
Телефон для справок: (095) 471-32-63
E-mail: editorial@computerpress.msk.su

Сдано в набор 01.02.93. Подписано к печати 15.02.93. Формат 84x108/16. Печать офсетная. Бумага типографская. Усл.печ.листов 8,4+0,42 (обложка). Кр.-отт. 10,08.Тираж 52000 экз. Заказ 3471. С-15.

Оригинал-макет подготовлен агентством «КомпьютерПресс».

Тексты проверены системой «ОРФО».

Отпечатано в полиграфической фирме «Красный пролетарий» РГИИЦ «Республика».
103473 Москва, И-473, Краснопролетарская, 16.



Известно, что чем выше профессиональный уровень программиста, тем шире он использует в своих программах динамическую память. В идеале программа в любой момент времени должна занимать в памяти минимум места. Создание и уничтожение массивов, нахождение компромисса между скоростью и размером — все это обязанность программиста, на плечи которого ложится нелегкая задача —

Работа с динамической памятью в Borland C++

Под динамической памятью применительно к Borland C++ будем понимать память, отличную от памяти, выделенной для хранения кода функций, статических данных, и не находящуюся в настоящий момент под управлением средств организации стека программы. Часто ее называют также "кучей" (heap). Важнейшей особенностью работы с динамической памятью является то, что управление ею полностью берет на себя программист.

Почти любая сколько-нибудь серьезная программа использует динамическую память. Это объясняется, главным образом, двумя обстоятельствами: во-первых, только при использовании динамической памяти возможно экономное и достаточно эффективное распределение (весьма ограниченных) ресурсов памяти ЭВМ и, во-вторых, поведение динамических объектов полностью определяется программистом; в частности, динамический объект, созданный внутри некоторой функции, не является локальным относительно этой функции и будет существовать и после ее завершения — до тех пор, пока он не будет удален из памяти явно. Особенно часто такая память используется для построения так называемых динамических структур данных — списков, очередей, хеш-таблиц.

Язык C не имел своей составной частью средств управления динамической памятью, и программисту приходилось использовать стандартные функции, такие, как malloc() и free(). Вообще говоря, отсутствие таких средств (как и отсутствие средств ввода-вывода) не считалось недостатком языка. Потребность в средствах управления памятью в C++ возникла, когда появилась необходимость единообразно управлять объектами как предопределенных, так и абстрактных типов, то есть типов, созданных самим программистом. Возможностей, предоставляемых функциями, подобными malloc(), оказалось явно недостаточно, и в C++ появились новые по сравнению с C операции — операция выделения динамической памяти new и операция освобождения динамической памяти delete. С появлением этих операций отпала всякая необходимость использования стандартных функций управления памятью C. Операции new и delete не только обладают большими возможностями, но и значительно удобнее.

Заметим, что операции new и delete являются единственными операциями C++, для выполнения которых привлекаются функции.

1. Работа с предопределенными типами данных

В этом разделе будет описано использование операций new и delete применительно к стандартным типам C++.

1.1. Операция new

Очень часто выражение, содержащее операцию new, имеет следующий вид (в угловые скобки помещен параметр, который может и отсутствовать):

```
указатель_на_тип = new имя_типа < (инициализатор) >
```

Имя_типа может быть любым, за исключением типа "функция"; тип "указатель на функцию" использовать можно.

Инициализатор — это необязательное инициализирующее выражение, которое может использоваться для всех типов, кроме массивов (напомним, что структура в C++ является частным случаем абстрактного типа и здесь пока не рассматривается).

Для начинающего программиста обратим внимание на следующее обстоятельство: при выполнении оператора

```
int *ip = new int;
```

создается два объекта: динамический безымянный объект и автоматический (или статический) указатель с именем ip, значением которого является адрес динамического объекта. Можно создать и другой указатель на тот же динамический объект, например, так:

```
int *other_p = ip;
```

С другой стороны, можно потерять доступ к нашему динамическому объекту, просто присвоив указателю ip другое значение:

```
int i;
//...
ip = &i;
```

В результате динамический объект, разумеется, будет существовать по-прежнему, но обратиться к нему уже невозможно.

Приведем простые примеры использования операции new:

```
int *ip = new int;                // создание динамического
                                  // объекта типа int
int *ip_init = new int(1);        // то же, с инициализацией
int *array = new int[10];         // создание динамического
                                  // массива
char **cp = new char*;            // создание динамического
                                  // указателя на char
char **cp_init = new char*(&c);   // то же, с инициализацией;
                                  // значением созданного ука-
                                  // зателя стал адрес пере-
                                  // менной  с  типа  char
double **d = new double*[j];      // создание динамического
                                  // массива указателей на
                                  // double; число элементов
                                  // в массиве равно j
```

В случае успешного завершения операция new возвращает указатель со значением, отличным от нуля. Результат операции, равный 0 (NULL), говорит о том, что не найден непрерывный свободный фрагмент динамической памяти нужного размера.

Следует обратить внимание на следующее обстоятельство: при выделении памяти и под одну переменную типа int, и под массив таких переменных операция new возвращает указатель на int (в случае использования массива этот указатель указывает на первый элемент созданного массива).

Очень удобно, что указатель, возвращаемый операцией new, уже имеет нужный тип, в то время как функция malloc() возвращает указатель типа void*; кроме того, может выполняться инициализация выделенного участка памяти (если задано инициализирующее выражение).

1.2. Обработка ошибок выделения динамической памяти

В ранних версиях C++ узнать, удачно или нет завершилась операция new, можно было только путем анализа значения возвращаемого указателя (при сравнении его со значением NULL). В версии C++ 2.0 можно явно задать функцию, которая будет автоматически вызываться в случае ошибки при выполнении операции new. Такая возможность поддерживается всеми компиляторами C++ фирмы Borland, даже Turbo C++ 1.x, в которой она почему-то не документирована. Покажем на простом примере, как это делается.

Небольшие предварительные пояснения. При использовании "малых" моделей памяти (tiny, small и medium) размер свободной динамической памяти можно явно задать с помощью присваивания некоторого значения глобальной переменной _heaplen, которая объявлена в файле dos.h, перед началом выполнения функции main(). Это значение будет задавать размер памяти в байтах. При работе с "большими" моделями памяти эта переменная не используется.

В нашей программе, которая должна компилироваться в одной из "малых" моделей, присваивание переменной _heaplen используется для того, чтобы операция new завершилась с ошибкой.

```
#include <iostream.h>
#include <new.h>        // Только для Borland C++ 2.x/3.x
#include <dos.h>

unsigned _heaplen = 10000;

void my_error(void);
void (*set_new_handler( void (*)() )) (); // не нужно в
                                          // Borland C++
                                          // 2.x/3.x

void (*old_handler)() = set_new_handler(my_error);

void main()
{
    char *cp = new char[20000];

    if (cp != NULL) cout << "Успех!\n";
    cout << "Конец работы программы.\n";
}

void my_error (void)
{
    cout << "Ошибка выделения динамической памяти!\n";
    set_new_handler(old_handler);
}
```

Результат работы программы будет выглядеть так:

```
Ошибка выделения динамической памяти!
Конец работы программы.
```

В нашем примере сначала происходит установка функции my_error() как функции обработки такой ошибки; при этом старое значение обработчика ошибки сохраняется как old_handler.

После появления "спровоцированной" ошибки вызывается функция my_error(), которая выводит сообщение на экран, а затем восстанавливает старый обработчик ошибки. Для чего это делается, вам станет ясно, если вы откомпилируете и запустите приведенную выше программу с таким вариантом функции my_error():

```
void my_error (void)
{
    cout << "Ошибка выделения динамической памяти!\n";
}
```

В этом случае ваша программа будет бесконечно выдавать сообщение об ошибке выделения памяти.

Обычно при появлении такого рода ошибки выполнение программы следует прекратить; для этого удобно функцию my_error() определить так:

```
#include <stdlib.h>

void my_error (void)
{
    cout << "Ошибка выделения динамической памяти!\n";
    exit(1);
}
```

1.3. Операция delete

Операция delete освобождает для дальнейшего использования в программе участок памяти, ранее выделенный операцией new. Синтаксис операции следующий:

```
delete указатель_на_тип;
delete [целое_выражение] указатель_на_тип;
```

указатель_на_тип — это указатель, возвращаемый как результат операции new.

Примеры использования операции delete:

```
delete ip;              // после ip = new int;
delete array;           // после array = new int[10];
delete[10] array;       // то же самое
```

Два последних примера иллюстрируют варианты применения операции delete при использовании динамических массивов. Оба варианта в данном случае будут работать совершенно одинаково. Дело в том, что при выделении памяти с помощью операции new фиксируется размер выделенного фрагмента памяти, и операция delete может его использовать для нормальной работы. В примере, подобном приведенному, явно указанный размер массива будет просто проигнорирован компилятором.

Тем не менее, существует случай, когда указывать размер динамического массива необходимо явным образом. Подробно это будет рассмотрено несколько позже.

В программах довольно часто встречается ситуация, когда операция delete более одного раза выполняется над одним и тем же указателем; обычно это следствие допущенной ошибки. Результат такого повторного применения в языке не определен: как правило, программа просто зацикливается. Тем не менее, бывают обстоятельства, когда такого использования операции delete избежать трудно. В таких случаях часто помогает то, что применение операции delete к указателю со значением NULL совершенно безопасно.

Начинающему программисту нужно ясно понимать, что операция delete работает только с динамической памятью, а не с указателем, который является ее операндом. После выполнения следующего фрагмента программы:

```
int *ip = new int;
delete ip;
```

указатель ip, естественно, продолжает существовать как самостоятельный объект; более того, и значение его остается тем же самым, которое он получил после выполнения операции new.

1.4. Функции, реализующие операции new и delete

Реализация операций с динамической памятью основана на использовании функций, имеющих имена operator new() и operator delete(). В языке предопределены две статические глобальные функции, имеющие вид:

```
void* operator new (size_t);
void  operator delete (void*, size_t = 0);
```

Имя size_t является синонимом типа unsigned int.

Выполнение операций с динамической памятью заключается не только в вызове соответствующих функций — как до, так и после вызова выполняются определенные действия. В частности, предварительно вычисляется длина требуемого участка в памяти (в байтах), и это значение передается функции operator new() в качестве ее первого аргумента; после же завершения функции выполняется преобразование результата ее работы — указателя типа void* — к указателю нужного типа. Кроме того, может выполняться инициализация выделенного участка памяти (если задано инициализирующее выражение).

Теперь ясно, какие ограничения накладываются на размер выделяемой памяти: вы не можете потребовать выделения области, размер которой превышает UINT_MAX байт (UINT_MAX — максимальное значение числа типа unsigned int); в Turbo C++ его значение равно 0xffff (65535). Вычисление требуемого размера в байтах, исходя из размера типа объектов и их числа, проводится по модулю UINT_MAX+1; например, в случае

```
int *ip = new int[32768];
```

значение первого аргумента при обращении к функции operator new() будет равно 0 (32768*sizeof(int) % (UINT_MAX+1) == 0).

Как и любые функции C++, функция operator new() является переопределяемой функцией, то есть программист может определить свои собственные варианты этой функции, отличающиеся числом или типами аргументов. Необходимо только удовлетворить определенным ограничениям, а именно: первый аргумент функции operator new() должен иметь тип unsigned, а возвращать она должна указатель на void.

В качестве примера рассмотрим программу, использующую переопределенную функцию operator new(), которая позволяет размещать создаваемый объект в определенном месте памяти (заметим, что применение такой функции позволяет создавать объекты, которые уже не являются динамическими и, следовательно, не могут быть удалены с помощью delete):

```
#include <iostream.h>
void* operator new (unsigned, void* addr);
int i = 0;
void main()
{
    int *ip = new(&i) int(5);
    cout << "Значение переменной i = " << i << endl;
}
void* operator new(unsigned size, void* address)
{
    return address;
}
```

Обратите внимание на то, как передается функции operator new() ее второй аргумент (адрес переменной i).

Если функция operator new() имеет больше двух аргументов, то после служебного слова new в скобках будет находиться не один аргумент, а их список, разделенный занятыми.

Функция operator delete() сильно отличается от operator new() тем, что если при создании динамического объекта программист знает о нем все (причем требования к такому объекту могут быть весьма разнообразны, и удовлетворение их может потребовать использования различных версий operator new()), то при уничтожении динамического объекта обычно известны лишь адрес и размер фрагмента памяти, который он занимает. Это приводит к тому, что нет никакой необходимости использовать большое число вариантов функции operator delete().

В отличие от operator new(), функция operator delete() может иметь только одну версию:

```
void operator delete (void*, size_t = 0);
```

Первый аргумент при вызове функции operator delete() инициализируется значением указателя, который является операндом операции delete.



Второй аргумент функции получает значение, отличное от значения по умолчанию, например, при удалении динамического массива данных, если операция delete имеет вид

```
delete[size] p;
```

Если p — указатель, например, на double, а size равен 10, то второй аргумент функции operator delete() будет равен 80 (10 * sizeof(double)).

В Borland C++ обе операции — и new, и delete — могут быть переопределены для конкретного абстрактного типа данных, к рассмотрению чего мы и переходим.

2. Операции new и delete при работе с абстрактными типами

С использованием в программе данных абстрактных типов связано наиболее творческое применение операций управления динамической памятью. Ранние версии C++ не поддерживали возможности переопределения стандартных функций operator new() и operator delete() для абстрактных типов данных — она появилась только в версии AT&T C++ 2.0. Для реализации этой возможности потребовалось внести в язык существенные изменения, а именно: разрешить использование статических функций — членов класса. Конечно же, это не значит, что в ранних версиях языка нельзя было создавать динамические данные абстрактных типов, но все-таки полное управление динамической памятью для своих собственных типов данных программист получил, начиная с версии 2.0. Все компиляторы C++ фирмы Borland поддерживают эту возможность.

Еще раз подчеркнем, что вне зависимости от того, описаны или нет функции operator new() и operator delete() как static, они всегда являются статическими функциями; из этого сразу вытекают два очень важных следствия: во-первых, эти функции не могут быть виртуальными функциями; во-вторых, для них не определен указатель this.

2.1. Операция new

Если абстрактный тип имеет конструктор без аргументов, то синтаксис обращения к операции new полностью совпадает с тем, что используется для выделения памяти под обычные типы данных без инициализирующего выражения:

```
class INT {
    int i;
};
void main ()
{
    INT* ptr = new INT;
   // ...
}
```

Если же конструктор класса INT имеет аргументы, то список аргументов помещается там же, где при ра-

боте со стандартными типами данных находится инициализирующее выражение. Сравните:

```
class INT {
    int i;
public:
    INT ();
    INT(int j) { i = j; }
};
void main()
{
    int *ip = new int(10);
    INT *lp = new INT(20);
```

По сути дела, в случае с абстрактными типами под именем типа понимается имя конструктора (которое совпадает с именем типа), а под инициализирующим выражением — список аргументов конструктора, то есть операция new в качестве своего операнда использует обращение к конструктору. Такая аналогия позволяет без труда ориентироваться в синтаксисе операции выделения памяти для абстрактных типов.

Если в операции new происходит обращение к конструктору без аргументов, то допустимы обе следующие формы записи:

```
INT* lp1 = new INT();
INT* lp2 = new INT;
```

Если конструктор без аргументов для класса X не определен, то при попытке выполнить оператор

```
X* xp = new X;
```

компилятор выдаст сообщение об ошибке (не найден конструктор X::X()). В этом случае требуется явно определить конструктор без аргументов.

В приведенных выше примерах использовалась стандартная операция new. Рассмотрим, как в C++ можно переопределить операцию new для заново создаваемого типа данных.

Как и при переопределении глобальной функции operator new(), переопределенная функция classX::operator new() должна возвращать результат типа void*, а ее первый аргумент должен иметь тип size_t.

Язык требует наличия первого аргумента типа size_t по очень простой причине: операция new, переопределенная для базового класса, наследуется производным от него классом:

```
class Base {
//...
    void operator new(size_t);
};
class Derived : public Base {
//...
};
Derived *dp = new Derived;
```

При выполнении последнего оператора будет вызвана функция Base::operator new() (если, конечно, операция new не была переопределена для класса Derived), но компилятор в качестве единственного аргумента при обращении к ней будет использовать размер объекта типа Derived, а не Base. Поскольку размеры объектов базового и производного типов в общем случае различны, задание размера в качестве аргумента совершенно необходимо для нормальной работы программы.

2.1.1. Использование new при создании единичного динамического объекта абстрактного типа

Как вы знаете, для любого объекта абстрактного типа для его инициализации используется конструктор. Что же происходит в приведенном ниже фрагменте программы?

```
class cl {
//...
public:
    cl (int arg) { /* ... */ }
    //...
};
//...
cl *cp = new cl(3);
```

Создание динамического объекта типа cl можно разбить на две стадии: во-первых, собственно создание объекта — это выполняет конструктор класса; во-вторых, размещение этого объекта в определенной области памяти — это делает операция new.

Взаимодействуют конструктор и функция operator new() достаточно сложным образом. С одной стороны, первой всегда должна выполняться функция operator new(): во-первых, конструктор сможет создать объект в уже выделенной под него памяти, во-вторых, в случае ошибочного завершения операции new (критерием чего служит возврат нулевого указателя) конструктор не следует вызывать вообще. С другой стороны, в версии 2.0 конструктор может сам разместить создаваемый объект в нужном месте, а следовательно, нет необходимости в обращении к функции operator new(), даже в том случае, если конструктор вызван из нее. Хотя размещение создаваемого объекта с помощью конструктора является устаревшей конструкцией языка и не рекомендуется к использованию по причине существенных недостатков, Borland C++ 1.x/2.0 (но не 3.x) ее поддерживает. Подробнее такой подход будет рассмотрен несколько позже.

Если конструктор сам не размещает создаваемый объект в памяти, то сначала выполняется функция operator new(), а затем код конструктора класса. Следующую программу мы советуем выполнить по шагам (тем, у кого есть такая возможность):

```
#include <iostream.h>
class cl {
public:
    cl ()  { cout << "Конструктор класса cl;" << endl; }
// Разрешено создание своей версии функции
// operator new(), которая будет использоваться
// только при работе с классом cl:
    void* operator new (size_t int);
};
void* cl::operator new (size_t size)
{
   cout << "Функция operator new() класса cl." << endl;
   return new char[size];  // не  return new cl !;
}
```

```
void main()
{
    cl *cp = new cl;
}
```

В результате работы программы получим такую последовательность сообщений:

```
Функция operator new() класса cl;
Конструктор класса cl.
```

В данном примере мы определили для класса cl свою, отличную от стандартной, версию функции operator new(), которая будет вызываться при работе с динамическими объектами только этого типа. Разумеется, мы можем определить в классе cl несколько вариантов функции cl::operator new(). Таким образом, программист может полностью взять на себя управление памятью при работе не только со стандартными, но и с абстрактными типами данных. При таком подходе объект, создаваемый с помощью операции new, может и не быть динамическим объектом в том смысле, что он может быть размещен в статической области памяти или в стеке.

В качестве небольшого упражнения замените в только что приведенном примере в функции cl::operator new() строку

```
return new char[size];
```

на

```
return NULL;
```

Каков будет результат работы такой программы?

2.1.2. Создание массива динамических объектов абстрактного типа

При создании динамического массива объектов абстрактных типов все происходит, как при создании "обычного" массива таких объектов — для каждого из элементов такого массива вызывается конструктор, не имеющий аргументов. Массив размещается в динамической памяти.

Рассмотрим пример:

```
#include <iostream.h>

class cl {
public:
    cl ()  { cout << "Конструктор класса cl;" << endl; }
    void* operator new (unsigned int);
    ~cl() {}
};

void* cl::operator new (unsigned size)
{
    cout << "Функция operator new() класса cl;" << endl;
    return (void*)((long)0x9000 << 16); // адрес 9000:0000
}

void main()
{
    cl *clp = new cl[3];      // массив из трех элементов
    cout << "Адрес начала массива = " << clp << endl;
}
```

Результат работы такой программы выглядит следующим образом:

```
Конструктор класса cl;
Конструктор класса cl;
Конструктор класса cl;
Адрес начала массива = 0x8f850e08
```

Обращения к функции cl::operator new() не происходит — мы работаем не с объектом типа cl, а с объектом "массив элементов типа cl", то есть cl*. Для этого типа операция new не переопределена (и не может быть переопределена, так как указатель всегда является стандартным типом), поэтому используется стандартная операция выделения памяти ::operator new().

Если вы хотите размещать такого рода массив явно, вам придется ввести новый тип — "массив элементов типа cl", и уже для него переопределить функцию operator new().

2.1.3. Использование конструктора для управления памятью

В программах, написанных с использованием Turbo/Borland C++, использовать конструктор таким образом не следует. Более того, подобный подход вообще запрещен в версии AT&T C++ 2.1, которую реализует компилятор Borland C++ 3.x. Жалеть об этом не стоит — использование конструктора так, как это будет показано ниже, не имеет никаких преимуществ по сравнению с рассмотренным ранее управлением памятью, зато имеет существенные недостатки. Тем не менее, имеет смысл кратко рассмотреть такой подход, так как он может встретиться в ранее написанных программах.

Использование конструктора для управления памятью основано на явном присвоении нужного значения указателю this.

При входе в конструктор в зависимости от класса памяти объекта указатель this имеет различные значения: если объект статический или автоматический, то this указывает на область памяти в сегменте данных или в стеке соответственно. Если же объект динамический, то есть конструктор вызывается из операции new, то при входе в конструктор значение указателя this равно 0 (NULL).

Если в конструкторе предусмотрено явное присваивание какого-либо значения указателю this, то это является признаком того, что управление динамической памятью для объектов данного типа полностью берет на себя программист, и нет никакой надобности в обращении к функции classX::operator new(); при этом компилятор никогда не проверяет, произошло ли на самом деле присваивание указателю this при выполнении конструктора или нет.

Одним из наиболее существенных недостатков такого подхода является необходимость выполнения присваивания указателю this на всех возможных путях выполнения тела конструктора. Рассмотрим пример:

```
class cl {
//...
public:
    cl (int i = 0, void* addr = 0)
        {
            if (this == NULL && i == 1)
```

```
            this = (cl*)addr;
         //...
      }
   //...
};
//...
cl *clp = new cl(i);    // стандартная операция new
```

В такой программе в зависимости от параметра i конструктора может происходить, а может и не происходить размещение создаваемого динамического объекта в определенном месте памяти; если значение аргумента i при обращении к конструктору отлично от 1, то значение указателя this останется неопределенным. В частности, при выполнении выражения

```
cl *cp = new cl(2, (void *)0x90000000);
```

не будет выполнено ни присваивание значения указателю this, ни вызвана операция new, то есть указатель cp получит значение NULL. Переопределение операции new для класса cl позволяет выполнить необходимые действия значительно удобнее:

```
void* cl::operator new(size_t size, int i = 0,
                       void *addr = 0)
{
   return i == 1 ? addr : new char[size];
}
```

2.2. Операция delete

При работе с данными абстрактных типов принципы использования операции delete те же, что и для операции new. Если при работе с динамическими объектами из операции new происходит вызов конструктора, то выполнение операции delete применительно к указателю на объект абстрактного типа приводит к вызову деструктора для этого объекта:

```
cl *clp = new cl(5); // вызов конструктора cl(5) после
                     // успешного завершения операции new
//...
delete clp;          // вызов деструктора ~cl() перед
                     // освобождением динамической памяти
```

Как и cl::operator new(), функция cl::operator delete() является статической функцией в классе cl, которая может быть переопределена.

При рассмотрении глобальной функции ::operator delete() в разделе 1.4 мы уже говорили о том, что нет никакой необходимости для операции delete иметь столько же разнообразных версий, как для операции new. Это полностью справедливо и для операции delete, переопределенной для некоторого класса X; функция X::operator delete() может иметь только две формы:

```
void operator delete (void*);
void operator delete (void*, size_t);
```

Если присутствует вторая форма данной операции, то компилятор использует именно ее. Обратите внимание, что смысл второго параметра функций ::operator delete() и classX::operator delete() различен: для функции — члена класса он задает размер объекта этого класса, насколько это известно компилятору, и используется для тех же целей, что и первый параметр функции X::operator new().

Необходимо обратить внимание на следующее обстоятельство: поскольку при использовании производных классов компилятор может и не знать истинный размер объектов (при работе с ними по указателям или ссылкам на базовый класс), то использование даже второй формы функции X::operator delete() может оказаться недостаточным. Для правильного удаления динамических объектов производного класса через указатель на их базовый класс совершенно необходимо использование виртуальных деструкторов.

Некоторые особенности операции delete связаны также с уничтожением динамических массивов объектов абстрактных типов с конструкторами и объектов, размещенных с помощью операции new, но не являющихся динамическими.

2.2.1. Удаление динамических массивов

Рассмотрим очень простой класс с достаточно искусственным конструктором:

```
const unsigned SIZE = 80;
class string {
   char *str;
   int size;
public:
   string ()          // конструктор без параметров
      { str = new char[size = SIZE]; }
   ~string()          // деструктор
      { delete str; }
   //...
};
```

При создании динамического массива объектов типа string, например, с помощью оператора

```
string *sp = new string[10];
```

10 раз произойдет вызов конструктора string::string(). Поскольку в конструкторе происходит "захват" динамической памяти, то перед освобождением участка памяти, занятого массивом, мы должны вызвать деструктор string::~string() для каждого элемента этого массива. Это можно выполнить с помощью оператора

```
delete[10] sp;
```

Здесь delete, разумеется, является стандартной операцией, а не операцией, переопределенной для класса string.

Удаление динамического массива элементов абстрактного типа с конструктором — единственный случай, когда необходимо явно указывать размер массива при обращении к операции delete.

Может возникнуть следующий вопрос: почему операция delete не может выполнить неявный цикл вызова деструкторов без дополнительного указания со стороны программиста — ведь вычислить число элементов в массиве, зная общий размер динамической области и размер типа, не составляет труда? Ответ на этот вопрос дан в книге автора C++ Б.Строструпа: управление объектами в динамической памяти — обязанность программиста, а не компилятора.

2.2.2. Явное обращение к деструктору

Следующее, на что следует обратить внимание — это как происходит удаление объектов абстрактных типов, размещенных с помощью операции new по явно указанному адресу.

```
class cl {
//...
public:
    cl()  { /* ... */ }
    ~cl() { /* ... */ }
    void* operator new (unsigned s, void* addr)
        { return addr; }
    //...
};

char *buffer[1000];

void main()
{
    cl *clp = new(buffer) cl;
    //...
}
```

В результате выполнения операции new объект типа cl будет размещен по начальному адресу глобального массива buffer, то есть в области статической памяти. Естественно, что уничтожать такой объект при помощи delete нельзя. Удаление такого объекта производится при помощи явного обращения к деструктору класса cl:

```
clp->cl::~cl();
```

При явном обращении к деструктору класса cl необходимо указывать его полное имя.

2.2.3. Использование деструктора для управления памятью

И наконец, осталось обсудить, как программист может отменить выполнение стандартной функции delete, явно работая в деструкторе с указателем this. Разумеется, что такой подход, как и явное присваивание некоторого значения указателю this в конструкторе, использовать теперь не надо.

При выполнении оператора

```
delete clp;     // clp -- указатель на объект класса cl
```

перед освобождением памяти, занятой динамическим объектом, будет вызван деструктор cl::~cl() для такого объекта. После завершения работы деструктора операция delete будет выполнена только в том случае, если значение указателя this отлично от 0. Таким образом, поместив в конце тела деструктора строку

```
this = 0;
```

программист отменит выполнение стандартной операции delete.



3. Пример использования переопределенной операции new

Механизм переопределения операции new нами уже был рассмотрен. Приведем пример, когда такой подход может оказаться чрезвычайно полезным.

Предположим, что для некоторой задачи удобно создать список динамических объектов типа stroka следующего вида:

```
struct stroka {
    char *str;          // указатель на строку
    stroka *nextElem;   // указатель на следующий элемент
                        // списка
};
```

Как видно, сам по себе объект типа stroka содержит только два указателя — на реальную символьную строку, которая, в свою очередь, может быть размещена в динамической памяти, и на следующий элемент типа stroka.

Такая организация хранения информации имеет много достоинств, но ей свойствен и определенный недостаток (который, правда, может и не рассматриваться как сколько-нибудь серьезный в определенном режиме работы — например, когда строки, помещаемые в список, вводятся оператором с клавиатуры). Суть его состоит в том, что при добавлении в список каждого нового элемента приходится дважды обращаться к операции выделения динамической памяти new, а это (в смысле затрат времени) далеко не безобидная операция:

```
//...
char *cp = new char[strlen(string)+1];
//...
stroka *sp = new stroka;
sp->str = cp;
```

С затратами времени на выделение динамической памяти собственно под символьную строку придется смириться — в общем случае все строки имеют различный размер, чего нельзя сказать об объектах типа stroka.

Разумно поступать следующим образом: выделять динамическую память блоками — сразу под несколько объектов типа stroka, и пока в блоке есть свободное место, размещать вновь создаваемые элементы типа stroka в нем.

Такой алгоритм тривиальным образом реализуется с помощью переопределения операции new для типа stroka:

```
struct stroka {
    //...
    void* operator new(size_t);
```

```
};
const NUMBER = 100;
void* stroka::operator new(size_t)
{
    static stroka huge * freeAddr = 0;
    static stroka huge * endAddr = 0;
    if (freeAddr == endAddr)    // выделение блока памяти
                                // под NUMBER элементов
    {
        freeAddr =
           (stroka huge*)new char[sizeof(stroka)*NUMBER];
        endAddr = freeAddr + NUMBER;
    }
    return freeAddr++;
}
```

Обратим внимание на некоторые детали.

Такой подход обеспечивает экономию не только времени, но и памяти ЭВМ — за счет того, что при выделении фрагмента динамической памяти резервируется место и для хранения размера самого фрагмента; ранее это происходило для каждого элемента типа stroka, теперь — только для каждого их блока.

Внутри функции operator new() при выделении блока динамической памяти мы не могли написать

```
freeAddr = new stroka[NUMBER];
```

так как это, очевидно, привело бы к бесконечной рекурсии вызовов функции stroka::operator new() самой себя.

И наконец, обратите внимание, что в определении функции stroka::operator new() в заголовке отсутствует имя единственного формального аргумента, хотя и присутствует его тип (size_t). Это имя отсутствует потому, что оно в дальнейшем не будет использоваться в функции; тип же присутствует для соответствия друг другу всех деклараций функции.

Мы обращаем на эту тонкость ваше внимание не для того, чтобы указать на не слишком интересную особенность синтаксиса языка C++. Рассмотренная функция stroka::operator new() — хороший пример того, как не надо писать программы. Отказавшись от использования параметра типа size_t и использовав вместо него при обращении к стандартной глобальной функции ::operator new() явно размер типа stroka, мы резко снизили универсальность написанной функции; теперь мы не можем использовать функцию stroka::operator new() для работы с классами, производными от класса stroka:

```
struct hashStroka : stroka {
    int hashValue;    // хеш-значение, сопоставленное со
                      // строкой
};
//...
hashStroka *hsp = new hashStroka;
```

Поскольку класс hashStroka является производным от класса stroka, он наследует его свойства, в том числе переопределенную для класса stroka операцию new.

Но функция stroka::operator new(), хотя и будет вызвана для типа hashStroka, будет работать неправильно — неверным будет затребованный размер динамической памяти.

Гораздо лучше определить функцию stroka::operator new() так:

```
void* stroka::operator new(size_t size)
{
    static char huge * freeAddr = 0;  // указатель на char,
    static char huge * endAddr = 0;   // а не на stroka !

    if (freeAddr == endAddr)        // выделение блока памяти
                                    // под NUMBER элементов
    {
        freeAddr = (char huge*)new char[size * NUMBER];
        endAddr = freeAddr + size * NUMBER;
    }

    void *retValue = freeAddr;
    freeAddr += size;
    return retValue;
}
```

Теперь значение параметра size будет вычислено компилятором именно для того типа, производного от типа stroka, для которого вызывается операция new; кроме того, правильно будет вычислено значение указателя endAddr.

Заметим, что возражение плана "я очень хорошо знаю, что никогда не буду использовать типы, производные от типа stroka" в данном случае свидетельствует о непрофессиональном подходе к программированию: если что-то можно сделать лучше и универсальнее (не теряя при этом ни в эффективности, ни в ясности программы), это должно быть сделано.

Но достоинства возможности переопределения операции new на этом не заканчиваются. Слабым местом приведенной программы является использование константы NUMBER. В самом деле, почему ее значение выбрано равным 100?

Эта проблема может быть разрешена чрезвычайно удобным и изящным образом, если переопределить функцию operator new() несколько по-другому, с использованием аргумента по умолчанию:

```
const NUMBER = 100;

struct stroka {
    //...
    void* operator new(size_t, int = NUMBER);
};

void* stroka::operator new(size_t size, int numb)
{
    static char huge * freeAddr = 0;
    static char huge * endAddr = 0;

    if (freeAddr == endAddr)
    {
        unsigned length = size * numb;
        freeAddr = (stroka huge*)new char[length];
        endAddr = freeAddr + length;
    }

    void *retValue = freeAddr;
    freeAddr += size;
    return retValue;
}
```

Возможность явного управления числом элементов в блоке важна еще и потому, что операция new не может выделить фрагмент памяти, размер которого превышает UINT_MAX.

Теперь при создании динамического блока объектов типа stroka при обращении к операции new можно использовать как значение по умолчанию:

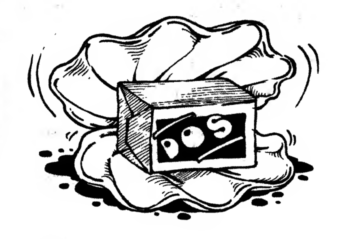
```
stroka *sp = new stroka;
```

так и явно задавать размер блока:

```
stroka *sp = new(200) stroka;
```

Надеемся, что материал этой статьи окажется полезным при написании нетривиальных программ.

А.Цимбал



DOSSHELL — окно в "Окна"

Как командовать программами

Аналог Windows Program Manager в Shell называется Program List, мы о нем уже упоминали. Разумеется, это далеко не Program Manager, но куда сильнее нортоновского User Menu!

В вашем каталоге программ уже есть несколько пунктов — выход в командную строку DOS, запуск редактора Edit, интерпретатора QBasic и несколько программ, объединенных в подкаталог дисковых утилит, а именно — "великие и ужасные" Backup и Restore, хорошо известные даже грудным детям Diskcopy и Format, а также новинки корпорации Microsoft — QuickFormat (старый Format с новым параметром /q) и программа восстановления стертых файлов Undelete (на самом деле — (C) Central Point Software). Возьмем сразу быка за рога и занесем в Program List что-нибудь свое, например, текстовый процессор *Лексикон*.

Вы уже приготовились к утомительному редактированию файла типа NC.MNU? Расслабьтесь.

Перейдем в окно каталога программ и выберем меню File|New. Shell спросит — будем делать Program Group (новый подкаталог программ) или Program Item (пункт выбора программы)? Выберем Program Group и в следующем предложенном нам диалоговом окошке (которое называется 'Add Group') укажем Title — Word Processors. В Help можно написать все, что вам заблагорассудится (можно даже явно указать места строчных разделителей — ^m), до 255 символов (в том числе и по-русски — к расширенным ASCII-кодам Shell относится совершенно нормально), в дальнейшем это сообщение будет появляться по F1. Если вы не желаете, чтобы кто-то другой пользовался на вашем компьютере редакторами текстов, можете заполнить Password. Кликнули 'OK' — и в Program List появился новый подкаталог Word Processors.

Окончание. Начало в КомпьютерПресс № 2'93.

Войдем в него и примемся за Program Item. Появится окно 'Add Program (рис. 8). Укажем Program Title для каталога программ — LEXICON (можно и так — ЛЕКСИКОН, только на нажатие первой буквы этот пункт нашего меню отзываться не будет). Как мы заполним Commands и Startup Directory — зависит от того, указан ли адрес *Лексикона* в переменной PATH, и в какой директории вы работаете с вашими документами. Можно Startup Directory и не заполнять. В Commands можно указать несколько команд подряд, разделяя их знаком ';' с пробелами по обе стороны, до 255 символов в строке. Если после выполнения всех команд не нужна пауза перед возвращением в Shell, выключите Pause After Exit. Можете и в этом окне указать Password. Если желательно предусмотреть под-

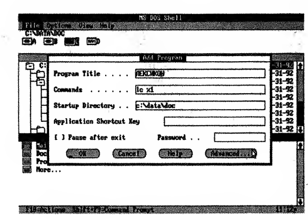


Рис. 8

сказку к вашему меню — выберите кнопку Advanced; можно набрать требуемый текст в первой строке появившегося окна (рис. 9) и, нажав 'OK', ввести его. Еще раз 'OK' — и в меню Word Processors появился ЛЕКСИКОН.

Кстати, было бы неплохо, если бы перед стартом *Лексикона* запрашивалось имя файла для редактирования. Можно было сделать это сразу, но и теперь не поздно. Выберите File|Properties, и на дисплее появится окно 'Program Item Properties', точно такое же, как и отредактированное вами 'Add Program (можно тем же путем отредактировать и уже имеющийся заголовок подкаталога программ — в окне 'Program Group Properties'). Надо дополнить команду вызова *Лексикона* замещаемым параметром '%1', как в BAT-файлах. Теперь на 'OK' Shell отзовется еще одним окном 'Program Item Properties , в котором вы укажете текст запроса на имя файла, и, возможно, имя файла, передаваемое *Лексикону* по умолчанию. Сколько вы укажете замещаемых параметров (можно до '%9'), столько надо будет заполнить дополнительных окон, и столько же запросов вы получите перед запуском *Лексикона*.

Можно создать сколько угодно пунктов меню, каталогов и подкаталогов и неограниченно их совершенствовать. Команды File|Copy, File|Reorder и DEL позволяют все созданное переупорядочивать и перемещать

Рис. 9

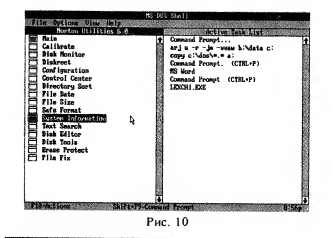


Рис. 10

(все изменения, сделанные вами в конфигурации Shell, заносятся в обычный текстовый файл dosshell.ini, который можно исправлять и вручную — редактором вроде Edit).

Таким образом, Shell поощряет вас к созданию пользовательских меню любой степени дружественности, разветвленности и, якобы, даже защищенности (конечно, все "защищенные" программы можно без всяких паролей вызвать непосредственно из командной строки или из File List). С хорошо разработанным Program List можно вообще забыть о "древесной" структуре жесткого диска. Нельзя не признать, что эта часть Shell сработана просто отменно.

Для завершения сеанса работы с Shell предусмотрена такая же, как и для выхода из Windows, комбинация клавиш — Alt+F4, но не спешите ее опробовать. Ведь нам предстоит познакомиться с САМОЙ очаровательной чертой нашей Shell, делающей ее настолько похожей на Windows, насколько вообще может младшая сестра походить на старшую. В руководствах Microsoft по DOS и Windows почти одинаковыми словами сказано следующее:

Вы можете выполнять несколько программ одновременно и легко переключаться между ними...

Действительно, используя встроенный в Shell переключатель задач (Task Swapper), вы можете переходить, например, из базы данных в электронную таблицу, а затем в редактор текстов или просто в командную строку одним-двумя нажатиями клавиш, не утруждая себя завершением работы с одной программой и загрузкой другой — в точности так же, как с Real и Standard modes Windows. И конечно, это переключение задач не имеет ничего общего с многозадачностью Windows 386 Enhanced mode. Да и не принесла бы много радости такая программно реализуемая возможность на медленной машине. Так что, по-видимому, единственным процессом, который действительно можно выполнять на машинах XT и AT одновременно с работой другой программы, остается фоновая печать через спулер.

А теперь, когда все точки над i в понятиях "многозадачность", "одновременность" и "переключение" расставлены, осталось разобраться, как же действует Task Swapper.

Включите Enable Task Swapper в меню Options. На экране (в режимах View|Program List или Program/File Lists) появится новое окно — Active Task List. Теперь после запуска любой программы можно нажать Ctrl+Esc — и через несколько секунд на экране появится все та же Shell. В окне Active Task List, изначально пустом, будет записано название запущенной программы или имя ее исполнимого файла, если вы загрузили его нажатием Enter из списка файлов, или командная строка, которую вы набрали в File|Run (рис. 10). Чтобы вернуться в программу, нажмите ее курсором или мышью — и вы застанете ее точно в том

же виде, в котором покинули. То же можно сделать, нажав Alt+Tab. Если в Active Task List находится более одной программы, то по Alt+Tab вы попадете в первую по списку. Непосредственно из программы можно попасть в следующую по списку, нажав Alt+Esc, а в предыдущую — нажав Shift+Alt+Esc. Но самый удобный способ переключения на нужную программу — нажимать Tab, удерживая Alt (на дисплее будут появляться названия программ, такие же и в том же порядке, как в списке активных задач Shell; выбрав нужное — отпустите Alt). Переключаться в обратном порядке можно, нажимая Tab и удерживая Shift+Alt.

Фактически такое количество shortcut key для переключения задач в Shell не нужно — вполне достаточно Ctrl+Esc для выхода в оболочку и Alt+Tab для циркулирования между активными программами. Включить программу в Active Task List, не загружая ее, можно, нажав Shift+Enter вместо Enter при запуске из File|Run, из списка файлов или списка программ. Точно такие же комбинации клавиш служат для переключения задач в Windows.

А если попытаться поместить в Active Task List несколько экземпляров одной и той же программы? Мы приучены к тому, что в одном каталоге не может быть двух одинаковых пунктов. Их и не будет — Shell добавит к "близнецу" точку, к следующему — еще одну точку и т.д.

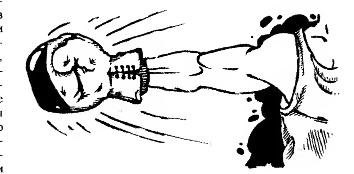
Некоторые дополнительные удобства предоставляет возможность входа в активные программы "из любого места" нажатием комбинаций клавиш, которые вы назначите этим программам при обустройстве своего меню пользовательских программ. Для этого надо в строке Application Shortcut Key нажать любое сочетание из Alt, Ctrl, Shift плюс любую буквенно-цифровую или функциональную клавишу. Фирма Microsoft предупреждает, что комбинации Ctrl и Shift+Ctrl с 'M', 'I', 'H', 'C', '[' и '5' (на цифровой клавиатуре) зарезервированы и в качестве клавиш вызова активных программ недоступны. Но — внимание! — на самом деле "запрещенных" комбинаций гораздо больше. Если вы определите для своих программ сочетания с Shift, то время от времени будете совершенно неожиданно "выпадать" из своего редактора текста при попытке написать заглавную букву, а сочетания с Alt не пройдут вам даром при работе в программах, где опции меню вызываются на Alt+букву — как и в самой Shell. В Microsoft Word зарезервированы для внутренних нужд чуть ли не все возможные комбинации клавиш, так что будьте бдительны!

Вот, собственно, и вся премудрость. Сам переключатель задач — это программа DOSSWAP.EXE, которая загружается перед выполнением любой не встроенной в Shell команды и заканчивает работу с возвращением в Shell, поэтому с пользовательской колокольни она кажется вполне резидентной. Работа с DOSSWAP вне Shell и запуск из командной строки не предусмотрены, поэтому в MS-DOS User's Guide and Reference о самом DOSSWAP нет ни слова. DOSSWAP занимает немногим более 30 Кбайт оперативной памяти, собственно DOSSHELL — чуть больше 4 байт (очень похоже на переключатель задач Windows DSWAP и резидент WIN).

Наверное, нет необходимости перечислять, какие конкретно удобства предоставляет возможность перехода между программами с сохранением контекста — даже если вы из осторожности не хотите совсем бросать свою работу на произвол судьбы и сохраняете результаты перед переходом в другие задачи. Однако всегда следует разбираться, какие обязанности накладывает на нас использование тех или иных прав.

Если у вас EGA-монитор, то обязательно установите из CONFIG.SYS драйвер EGA.SYS, входящий и в DOS, и в Windows. Именно этот драйвер сохраняет и воспроизводит изображение на EGA. "Счастливцам", работающим на CGA, следует устанавливать в окне Advanced редактора Program List режим Video Mode — Graphics для программ, которые идут в графическом режиме. Всем остальным этот режим потребуется при переключении программ, рассчитанных для CGA. Я с удивлением обнаружил, что в игре "F19" на EGA после переключения задач портится изображение, однако если программе при загрузке указать режим адаптера CGA (и Video Mode — Graphics), то все идет, как полагается — с четырьмя цветами. Знаменитый "Принц" не позволяет переключать себя на EGA, но на VGA — не сопротивляется.

Когда вы переключаетесь из одной программы в другую или в Shell, Task Swapper создает два файла с расширением 'TMP': один — на 8 Кбайт, в директории, где находятся файлы Shell; другой — от сотни до нескольких сот килобайт, с атрибутами "скрытый" и



"только для чтения", в директории, указанной в переменной окружения TEMP. Именно в этом файле сохраняется вся обстановка системы на момент переключения программы. Task Swapper не позволит, насколько это в его силах, чему-либо измениться в оперативной памяти — если вы загрузите в командной строке TSR-программу, то перед возвращением в Shell вам будет в ультимативной форме предложено ее выгрузить нажатием Ctrl+C. Если же вы возвращаетесь в Shell не по команде 'exit', а через Ctrl+Esc, то переключатель задач, не мудрствуя лукаво, сохранит наличие TSR в своем TMP-образе, а из памяти уберет; но вы снова сможете пользоваться вашей ненаглядной резидентной программой, вернувшись в командную строку!

Таким образом, если в целях экономии памяти вы не хотите держать все резиденты вместе, то используйте несколько независимых командных строк с разной окружающей средой — в одной работает дискетный драйвер 800, в другой — драйвер экрана и клавиатуры RK, в третьей вы рисуете картинки с помощью Capture... Серьезная проблема встречена мною при работе с версией 1 пакета *Русское Слово*, а именно — с резидентом RUSS, совместная деятельность которого с DOSSWAP ни к чему хорошему не приводит. В случае появления подобных проблем с программами, без которых вы жить не можете, включайте для них Prevent Program Switch в окне Advanced редактора Program List.

Учтите, что Task Swapper резервирует для своих целей системные хендлы, и при использовании DOSSWAP у вас фактически будет меньше доступных файловых каналов, чем указано в строке FILES вашего CONFIG.SYS, что может не понравиться таким пакетам, как Harvard Graphics 3.0 или Paradox.

Раскрутите ваш винчестер

Вместе с привлекательной возможностью достаточно фривольного обращения с оперативной памятью появляются особые требования к состоянию жесткого диска, ибо любые изменения в структуре данных на нем могут повлечь потерю информации и порчу активных задач — понятно почему (если непонятно, см. предпредпоследний абзац предыдущего раздела). Отсюда мораль — не трогайте файлы, непосредственно используемые в ваших активных задачах, вне этих задач (в основном это касается не тех программ, которые полностью "всасывают" все, что им необходимо, в оперативную память, а тех, которые постоянно обмениваются данными с диском, как это делают системы управления базами данных) и, уж конечно, не прикасайтесь к самим временным файлам, которые делает Task Swapper. Кроме того, не допускается обработка винчестера "докторами" и оптимизаторами дисков. Советую делать это из AUTOEXEC.BAT (лишняя минута загрузки на самом деле только сэкономит вам время, которое неминуемо придется потратить, когда на диске образуется с мегабайт потерянных и перекрещенных цепей, а остальное пространство будет процентов эдак на 50 фрагментировано). Task Swapper "знает в лицо" CHKDSK/F, FASTOPEN, SHARE и не разрешит вам их запустить.

Становится понятно, что успех работы с Shell зависит в первую очередь от того, в каком состоянии ваш винчестер — достаточно ли на нем места для своппинга текущих задач, исправны ли его логическая и физическая структуры, хорошо ли он оптимизирован и т.п. Можно попытаться назначить переменную TEMP на электронный диск, однако Shell может не принять этого всерьез, если предоставленный для своппинга объем покажется ей недостаточным, или же съехать на жесткий диск, когда RAM-диск окажется заполненным. Собственно, речь здесь идет уже о вопросах оптимальной конфигурации системы, которые важны ВСЕГДА; однако при переключении задач, когда для перехода из программы в программу необходимо записать один большой файл и прочитать другой, скорость доступа к диску приобретает ярко выраженную идеологическую окраску. В рамках данной статьи было бы неразумно пытаться что-то добавить к горам бумаги, посвященным способам улучшения работы жесткого диска, тем более что недавно издательство "Мир" выпустило книгу "Работа с жестким диском IBM PC" Питера Нортона и Роберта Джордейна (первый известен всем и в рекомендациях не нуждается, второго знают те, кто пользуется его "Справочником программиста IBM PC, XT и AT"), и к написанному ими еще долго нечего будет добавить...

Отправной точкой данной статьи была проблема подбора подходящего программного обеспечения для компьютеров разного класса. На 386-м вы, конечно, будете работать в среде Windows. Пока живы машины XT, на них будет верой и правдой служить Norton Commander. А пользователям IBM AT остается пожелать приятного знакомства и плодотворного сотрудничества с MS-DOS Shell.

К.Ахметов

Использование динамических виртуальных методов в Turbo Vision

Введение динамических виртуальных методов в реализацию языка Pascal (рассматривается компилятор Turbo Pascal фирмы Borland) преследовало, по-видимому, две цели:

- обеспечить механизм обработки сообщений Windows в реализации объектно-ориентированной библиотеки ObjectWindows;
- сократить объем памяти, занимаемой таблицами виртуальных методов в сегменте данных программы.

Поддержка динамических виртуальных методов компилятором Turbo Pascal 7.0 для DOS открывает возможность использования таких методов в DOS-приложениях. Один из очевидных вариантов использования таких методов — это повторение механизма вызова методов обработчиков сообщений, используемого в библиотеке ObjectWindows, для библиотеки Turbo Vision. Рассмотрим, каким образом это можно сделать.

Прежде всего необходимо иметь процедуру, которая позволяет определить исполнительный адрес метода по индексу. Пример такой процедуры приводится ниже. В качестве параметра функции Dynamic передается адрес таблицы виртуальных методов (VMT) объекта, у которого нужно найти адрес метода по индексу и сам индекс. Возвращаемое значение содержит исполнительный адрес метода или Nil, если метод с заданным индексом не найден.

```
Function Dynamic(VMT: pointer; Index: word): pointer; assembler;
asm
     std                        {сканирование назад}
     mov   ax,[Index]  {загрузка индекса поиска}
     les   di,[VMT]             {es:di указатель на VMT}
     mov   di,es:[di+4]         { указатель на DMT, если имеется}
@Check:    or    di,di {проверка наличия DMT}
     jnz   @Exists
     mov   ax,di        {Нет DMT:  Возврат NIL}
     mov   dx,di
     jmp   @Exit
@Exists:
     mov   cx,es:[di+6]         {длина VMT }
     mov   bx,cx
     shl   bx,1                 {удвоение величины для адресации}
     mov   si,di        {восстановление указателя на DMT}
     lea   di,[di+bx+6]         {смещение последнего индекса}
     repne scasw {поиск нужного}
     jne   @NoMatch
@Match:    shl   cx,1           {удвоение найденного индекса}
     shl   cx,1
     add   bx,cx        {добавить к начальному}
     les   ax,[es:si+bx+8]{@DMT + SizeOf(Indices) +
                          SizeOf(Hdr) +
                          MatchIndex*SizeOf(pointer)}
     mov   dx,es
     jmp   @Exit
@NoMatch:
     mov   di,es:[si]  {смещение родительской DMT}
     jmp   @Check
@Exit:
end;
```

Если имеется исходный текст библиотеки ObjectWindows, то вариант такой процедуры можно использовать и оттуда.

Теперь рассмотрим один из вариантов реализации вызова методов обработчиков сообщений. Пусть необходимо обеспечить обработку команд в объекте — наследнике объекта TApplication при помощи динамических виртуальных методов. Определение такого объекта может выглядеть следующим образом:

```
TTutorApp = object(TApplication)
ClipboardWindow: PEditWindow;
OrderWindow: POrderWindow;
constructor Init;
destructor Done; virtual;
procedure DoAboutBox(var Event: TEvent); virtual cmAbout;
procedure HandleEvent(var Event: TEvent); virtual;
procedure InitMenuBar; virtual;
procedure InitStatusLine; virtual;
procedure Commands(var Event: TEvent); virtual;
procedure LoadDesktop(var Event: TEvent); virtual cmOptionsLoad;
procedure NewWindow(var Event: TEvent); virtual cmNew;
procedure OpenOrderWindow(var Event: TEvent); virtual cmOrderWin;
procedure OpenWindow(var Event: TEvent); virtual cmOpen;
procedure SaveDesktop(var Event: TEvent); virtual cmOptionsSave;
end;
```

Отметим, что за основу был взят пример номер 8 из поставки Borland Pascal 7.0. Все методы — обработчи-

ки командных сообщений являются динамическими виртуальными методами и имеют индекс, соответствующий величине кода команды, и не имеют каких-либо особенностей. Изменения коснулись метода HandleEvent, который теперь имеет следующий вид:

```
procedure TTutorApp HandleEvent(var Event  TEvent),
var
R  TRect;
begin
Inherited HandleEvent(Fvent),
Commands(Event);
  if Event What = evCommand then
  begin
    case Event.Command of
      cmClipShow
        with ClipboardWindow^ do
        begin
          Select,
          Show;
          ClearEvent(Event),
        end,
      cmOptionsVideo
        begin
          SetScreenMode(ScreenMode xor smFont8x8),
          ClearEvent(Event),
        end,
    end,
  end;
end;
```

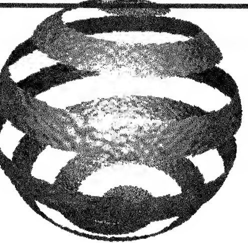
Как видно из текста, обработчик сообщений существенно не изменился, добавлен лишь вызов нового метода Commands. Этот метод и вызывает соответствующий метод — обработчик сообщения по его индексу, являющемуся кодом команды. Текст этого метода следующий:

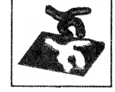
```
TEventProc = procedure(var Event  TEvent),

procedure TTutorApp Commands(var Fvent  TEvent);
var
  EventProc  TEventProc,
  ProcAdr  Pointer absolute EventProc;
begin
   if Event What <> evCommand then Exit;
   ProcAdr  = Dynamic(TypeOf(Self), Event Command),
   if ProcAdr = Nil then Exit;
   EventProc(Event)
   ClearEvent(Event)
end
```

Те, кто обладает исходным текстом библиотеки Turbo Vision, могут встроить этот метод в ядро библиотеки и таким образом придать ей новое качество.

Д.Рогаткин









Мы завершаем публикацию глав из книги Д.Рогаткина и А.Федорова "Turbo Pascal в среде Windows". Начало — в №№ 6, 8, 9, 11, 12'92. В этом номере продолжается разговор о ресурсах — специальном типе данных, используемом в Windows для отображения графической информации.

TURBO PASCAL в среде WINDOWS

Шаблон заполнения фона

Растровое изображение может быть использовано в качестве шаблона заполнения фона рабочей области экрана. После того как этот шаблон создан, он выбирается и копируется на экран функцией PatBlt.

```
Function PatBlt(DC : hDC; X, Y, Width, Height: Integer; Rop: LongInt);
```

Функция используется для выполнения графических операций над уже имеющимся шаблоном для заполнения фона.

При заполнении экрана указанный шаблон копируется необходимое число раз по горизонтали и вертикали. Размер растрового изображения при заполнении фона не меняется (не уменьшается и не увеличивается).

Обычно для заполнения фона используется изображение размером 8x8 пикселов. Изображение может быть и большего размера, но для заполнения экрана будет использована только его часть размером 8x8 пикселов, остальное изображение игнорируется. В любом случае после получения ссылки на изображение шаблон заполнения фона может быть создан функцией CreatePatternBrush.

```
Function CreatePatternBrush(Bitmap : hBitmap): hBrush;
```

Функция используется для создания кисти с указанным рисунком. В качестве параметра указывается идентификатор растрового изображения. Функция возвращает ссылку на созданную кисть.

Ниже показано, как загрузить растровое изображение, хранимое в виде ресурса, и создать с помощью него шаблон заполнения фона.

```
MyBitmap := LoadBitmap(hInstance,'MyPattern');
MyBrush := CreatePatternBrush(MyBitmap);
```

После создания шаблона он включается в контекст:

```
hOldBrush := SelectObject(MyDC,MyBrush);
```

Выбрав шаблон заполнения фона, указанную часть окна заполняют с помощью функции PatBlt. Ниже приводится пример заполнения левого верхнего угла экрана с помощью ранее загруженного и выбранного шаблона:

```
PatBlt(MyDC,0,0,100,100,PatCopy);
```

Тип операции PatCopy указывает на необходимость перерисовки фона по указанному шаблону.

Шаблон заполнения фона окна можно также использовать при создании класса окна, указав его в поле hbrBackground структуры типа TWindowClass:

```
WndClass.hbrBackground := CreatePatternBrush(MyBitmap);
```

После этого каждый раз при перерисовке фона будет использоваться указанный шаблон заполнения фона окна. Можно изменить шаблон, прибегнув к уже знакомой функции SetClassWord:

```
SetClassWord(MyWnd,gcw_hbrBackground,MyBrush);
```

Отметим, что вызов данной функции изменит шаблон заполнения фона для всех окон данного класса. При необходимости изменения шаблона фона лишь одного окна обрабатывают сообщение wm_EraseBkgnd:

```
Var
  MyRect : TRect;
  MyOldBrush : hBrush;
  ....
  ....
```

Таблица 2. Функции ядра Windows, предназначенные для манипулирования палитрами

Функция	Описание
AnimatePalette	Заместить элементы логической палитры. Новые элементы помещаются в системную палитру
CreatePalette	Создать логическую палитру
GetNearestPaletteIndex	Определить индекс элемента логической палитры для указанной RGB-величины
GetPaletteEntries	Определить элементы логической палитры
GetSystemPaletteEntries	Определить элементы системной палитры
GetSystemPaletteUse	Определить, имеет ли прикладная программа доступ ко всей системной палитре
RealizePalette	Отождествить элементы логической палитры с элементами системной палитры
SelectPalette	Выбрать палитру для контекста устройства
SetPaletteEntries	Создать новые элементы в логической палитре
SetSystemPaletteUse	Предоставить прикладной программе полный доступ к системной палитре
UpdateColors	Выполнить попикселное преобразование цветов на основе значений системной палитры

Таблица 3. Функции работы со шрифтами

Функция	Описание
AddFontResource	Добавить ресурс из указанного файла в таблицу системных шрифтов
CreateFont	Создать логический шрифт с указанными характеристиками
CreateFontIndirec	Создать логический шрифт с указанными характеристиками
EnumFonts	Перебрать шрифты, доступные данному устройству
GetCharWidth	Определить ширину указанного символа
RemoveFontResource	Удалить шрифт из таблицы системных шрифтов

```
Case Message OF
  ....
  wm_EraseBkgnd :
    Begin
     UnrealizeObject(MyBkgndBrush);
     MyOldBrush := SelectObject(wParam,MyBkgndBrush);
     GetUpdateRect(wParam,MyRect,False);
     PatBlt(wParam,MyRect.Left,MyRect.Top,
      MyRect.Right-MyRect.Left, MyRect.Bottom-
MyRect.Top,PatCopy);
     SelectObject(wParam,MyOldBrush);
    End;
```

Параметр wParam сообщения wm_EraseBkgnd содержит ссылку на контекст экрана. Функция SelectObject помещает необходимый шаблон в контекст экрана. Функция GetUpdateRect вычисляет область, фон которой необходимо перерисовать. Функция PatBlt выполняет непосредственную перерисовку области окна. Далее происходит восстановление предыдущего шаблона заполнения фона.

Палитры

Цветные графические дисплеи способны отображать огромное количество цветов. Однако в большинстве



случаев число одновременно отображаемых цветов ограничено, например, видеоадаптер VGA способен отображать лишь 256 цветов. Набор цветов, одновременно отображаемых на экране, называется палитрой. Когда прикладной программе требуется отсутствующий цвет, он добавляется в палитру. Если число одновременно отображаемых цветов превышает аппаратно допустимый максимум, один из элементов палитры замещается требуемым. Таким образом, если общее количество цветов, используемых одним или несколькими окнами, превышает максимально допустимое, это сказывается на цветовом представлении других окон.

Ядро Windows содержит ряд функций (табл. 2), позволяющих манипулировать палитрами.

Установка палитры

Для создания логической палитры используется функция CreatePalette, которой передается указатель на структуру данных типа TLogPalette. Эта функция



создает палитру и возвращает ссылку, используемую при вызове функции SelectPalette, позволяющей включить палитру в контекст устройства.

Структура TLogPalette содержит поля, описывающие номер версии Windows ($0300), число элементов в палитре и массив этих элементов. Каждый элемент палитры описывается записью типа TPaletteEntry. Тип данных TPaletteEntry содержит три поля, указывающие вес цветовой составляющей элемента палитры: peRed, peGreen и peBlue и поле флагов peFlags.

Вызов функции GetStockObject(Default_Palette) создает палитру, содержащую 20 цветов, входящих в системную палитру.

После того как прикладная программа включила палитру в контекст устройства, вызвав функцию SetPalette, перед использованием этой палитры необходимо произвести следующие действия:

```
MyPalette := CreatePalette(@MyLogPalette);
SelectPalette(MyDC,MyPalette,0);
RealizePalette(MyDC);
```

Функция RealizePalette отождествляет цвета палитры MyPalette с цветами системной палитры устройства. Сначала выполняется поиск цветов, совпадающих в обеих палитрах, затем недостающие цвета добавляются в системную палитру. Наконец, цвета, соответствие которым не найдено в системной палитре, замещаются на наиболее близкие цвета системной палитры (по весовым коэффициентам составляющих).

Использование палитры

Установив палитру, прикладная программа начинает использовать ее цвета. Имеется возможность прямого и косвенного указания цветов. При прямом указании цвета используется индекс TColorRef. Это число типа LongInt, старший байт которого установлен в 1, а младший содержит номер элемента палитры. Например, число $0100000A указывает на 10-й элемент палитры. Оно может использоваться везде, где используется тип данных TColorRef.

Изменение палитры

Изменение элементов палитры и отображение этих изменений на экране выполняет функция AnimatePalette. Ниже приводится пример изменения значений первых десяти элементов палитры с отображением этих изменений:

```
GetObject(MyPalette,SizeOf(NumEntries), @NumEntries);
if NumEntries >= 10 then
begin
  GetPaletteEntries(MyPalette,0,10,@PaletteEntries);
  for I := 0 to 9 do
  begin
    PaletteEntries[I].peRed := PaletteEntries[I].peRed + 40;
    PaletteEntries[I].peGreen := PaletteEntries[I].peGreen - 40;
    PaletteEntries[I].peBlue := PaletteEntries[I].peBlue - 40;
  end;
  AnimatePalette(MyPalette,0,10,@PaletteEntries);
end;
```

Вместо вызова AnimatePalette можно задать:

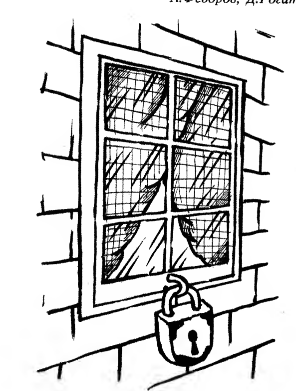
```
SetPaletteEwntries(MyPalette,0,10,@PaletteEntries);
RealizePalette(MyPalette);
```

Шрифты

Модуль GDI содержит ряд функций выбора, создания, удаления шрифтов, а также получения информации о них. Для работы со шрифтами используют функции, приведенные в табл. 3.

Более подробно процесс создания и использования шрифтов рассматривается в разделе, посвященном отображению символьной информации.

А.Федоров, Д.Рогаткин





Проблема автоматического чтения текстов не нова, и ей ежегодно посвящается немало обзоров. Поднимавшийся раньше вопрос, а нужны ли сейчас и в будущем читающие системы, уже не дискутируется. Обсуждение переместилось в другую плоскость: что же собственно достигнуто и какими мы хотели бы видеть эти системы?

Что такое системы автоматического чтения текстов?

или Еще раз про OCR

Анализируя сегодняшнее состояние OCR-систем (OCR — optical character reading), мы не приводим детального сравнения различных разработок, которое можно найти в различных журналах. Основное внимание сосредоточено на наиболее важных показателях читающих систем, характеризующих их сегодняшний уровень, и на определении путей их развития с тем, чтобы они действительно могли стать повседневным инструментом в работе пользователей ЭВМ и взяли бы на себя какую-то часть функций наших глаз, о чем еще в 50-х годах говорил Фрэнк Розенблат, один из основателей нейрокибернетики. Что представляют собой сегодняшние читающие системы? Для сегодняшнего состояния систем автоматического чтения наиболее характерными являются следующие показатели.

Надежность распознавания

Эта характеристика является ключевой в оценке системы. В конечном итоге именно она определяет не только класс системы, но и уровень развития технологии автоматического чтения в целом. Этот показатель может быть очень высок — одна ошибка на десятки или даже сотни тысяч знаков, но при этом требуется исключительно высокое качество печати. На реальном, а не на специально подготовленном материале ошибок значительно больше. Так как сегодня не существует хорошо отработанных формальных показателей качества печати текста, сравнение надежности различных систем основано на некотором среднем по качеству материале и носит достаточно условный характер. Наилучший способ сравнения систем — на одном и том же материале.

Когда называют процент правильно распознанных символов, обычно предлагают один показатель надежности. Для лучших систем это обычно 99,5% и выше. Однако эта характеристика требует уточнения. Следует называть два параметра: вероятность отказа (reject) и вероятность ошибки (substitution, mistake). Отказ регистрирует сама система распознавания, и он может быть автоматически обработан контекстным процессором или оператором. Ошибка имеет более тяжелые последствия: она или вообще может оказаться незамеченной, или потребует значительно большего времени на исправление. Но всегда можно, увеличивая число отказов, уменьшить число ошибок (естественно, в разумных пределах).

Средства для реализации читающих систем

За небольшим исключением, современные читающие системы реализованы чисто программными средствами с использованием стандартных персональных компьютеров типа IBM PC или Macintosh и настольных сканеров. Наиболее мощные программы могут быть реализованы только на PC 386 или Macintosh, имеющих ОЗУ объемом не менее 2–4 Мбайт. Некоторые системы работают и на менее мощных машинах, но требуют установки специальных плат расширения.

Настольные сканеры могут быть либо автоматическими, либо ручными. Большинство программ работает с автоматическими сканерами. Среди них различают типы flat bed и sheet fed. В первом сканируется неподвижный лист, а движется считывающая система, во втором — протягивается лист мимо неподвижной считывающей системы.

Сканеры flat bed заметно дороже, но они позволяют считывать книги и журналы без разделения их на отдельные листы.

Ручные сканеры позволяют считывать полосы шириной 4 или 8 дюймов и записывать информацию на диск. Ввод больших массивов документов неудобен, так как требуется постоянное участие человека. Кроме того, скорость движения рукой должна быть строго дозирована, при вводе могут быть сильные перекосы. При использовании 4-дюймовых сканеров считывание наиболее распространенных форматов A4 требует дальнейшего "склеивания" введенных изображений (или распознанных текстов). Ручные сканеры удобны при выборочном чтении узких газетных или журнальных полос.

В последнее время появился сканер комбинированного типа, сочетающий свойства авто- и ручного сканера. Он использует протяженный фоточувствительный прибор, совмещенный с осветителем, благодаря чему имеет небольшие габариты, но способен обрабатывать листы шириной 8 дюймов. Сканер содержит специальную подставку для протягивания бумаги, но считывающая головка может сниматься, и тогда сканер превращается в ручной.

Существует еще один вид автоматического сканера, конструктивно напоминающего фотоувеличитель. Он обеспечивает легкое манипулирование страницами.

Большинство сканеров имеет разрешение 300 точек на дюйм, но некоторые из них, в том числе ручные, обладают повышенным разрешением — 400 точек на дюйм. Типичное количество градаций видеосигнала — 16, 64 и 256, но в режиме считывания текста требуется всего 2 уровня.

Скорость ввода

Типичная скорость ввода с помощью программных пакетов, реализованных на ЭВМ типа PC AT,— от 0,5 до 2 минут на страницу. Разумеется, применяя более мощные машины, можно автоматически увеличивать скорость, но при этом не следует забывать о резко возрастающей цене аппаратуры. Для систем с дополнительной аппаратной поддержкой это время может быть уменьшено до 10 секунд на страницу.

Обучение новым шрифтам

Одна из важнейших характеристик читающих (и вообще распознающих) систем — способность к обучению. По этому признаку все читающие системы часто делят на обучаемые и необучаемые. Какую систему предпочесть?

Ответ на этот вопрос не так очевиден, как кажется на первый взгляд. Действительно, вполне понятно, что желательно иметь возможность обучения, но почему же тогда лучшие OCR-пакеты, называемые даже интеллектуальными, являются необучаемыми. Ответ очень простой — потому что интеллект этого пакета — это интеллект разработчика, тщательно составляющего описание символов, а обучаемые системы лишены такой изощренности, и само обучение носит довольно прямолинейный характер. Особенно это сказывается, когда безуспешно пытаются обучить распознаванию близких по форме символов. Ниже мы еще вернемся к этому вопросу и рассмотрим свой подход к проблеме. Однако в целом на сегодня ситуация именно такова — лучшие по надежности системы являются необучаемыми, а описание форм наиболее употребительных шрифтов уже заранее введено в программу.

Количество шрифтов

Вопрос о количестве шрифтов — один из самых главных. Сегодня, пожалуй, лишь единичные пользователи могут ограничиться одним шрифтом. Среди многошрифтовых систем различаются omnifont и multifont. Системы omnifont многошрифтовые по существу и могут обрабатывать тексты различных шрифтов на одной странице. Системы multifont по принципу своего действия являются одношрифтовыми, и перед

Настройка сканера

Настройка сканера на документ включает установку ряда параметров — геометрической разрешающей способности, размеров поля считывания, контраста печати. Наиболее трудоемкой операцией является оценка и установка контраста. Авторы обзоров обычно не уделяют этой проблеме особого внимания, так как для многих зарубежных изданий с высоким качеством печати это достаточно простая процедура. Но для текстов, где уровень контраста визуально оценить трудно, и тем более, если документы при относительно небольшом объеме каждого из них часто меняются из-за того, что они получены из различных источников, такая работа занимает значительное время. Эти трудности обусловлены как недостатками в программном обеспечении, так и особенностями работы самих сканеров, которые не позволяют в процессе сканирования одной страницы менять программным способом установку контраста. Поэтому приходится осуществлять многократный ввод, оценивать качество этого ввода сначала визуально, а затем и окончательно в процессе самого распознавания.

Только в последнее время появилась технология автоматической установки контрастов, реализованная в новом сканере HP ScanJet IIP. Однако пока нет надежного подтверждения эффективности ее использования при чтении низкокачественных текстов.

распознаванием определенного типа шрифта они должны загружать с диска в ОЗУ его описание. Эта операция делается либо автоматически, либо по указанию оператора. При чтении текстов с разнообразными шрифтами системы multifont мало эффективны.

Разметка или сегментация страницы

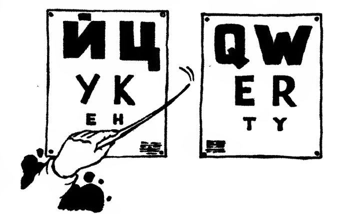
Если еще несколько лет назад такую возможность имели лишь самые дорогие системы, то сегодня это обычный атрибут. Разнообразие вводимого материала требует разбиения страницы на текстовые и графические зоны. Кроме того, сама текстовая часть может иметь сложную структуру и состоять из многих колонок. Разбиение на зоны может производиться как интерактивным, так и автоматическим способом. Однако здесь следует обратить внимание на то, что такая операция, в особенности автоматическая, без дополнительного объема ОЗУ (то есть свыше 640 Кбайт) может оказаться чрезвычайно неэффективной по времени. Другие ограничения связаны с трудностями разбиения на зоны очень сложных по формату данных (например, если зоны не имеют прямоугольной формы) или при сильном перекосе страницы. Операция устранения перекоса будет более эффективной при физическом выравнивании листа и его повторном вводе, чем при корректном повороте всей страницы программным способом.

Контекстная обработка

Совершенно очевидно, что надежное чтение текстов на уровне только символьного распознавания возможно лишь при очень высоком качестве текстов. Реально таких документов очень мало. Естественным способом повышения надежности является контекстная обработка. Наиболее употребительный контекстный процессор — это spelling checker, который обрабатывает тексты по правилам орфографии. В отличие от стандартных программ такого рода, с OCR-системами работают их специальные модификации, которые учитывают особенности автоматического распознавания текста. Такая модификация состоит в том, что контекстному процессору сообщается информация о близких по начертанию классах, о гипотезах или о претендентах на правильный ответ и т.д. Как пример можно привести буквы **с** и **е**. Буква **е** очень часто при вводе теряет свою среднюю горизонтальную часть, и эта ошибка может быть обнаружена только на уровне контекста.

Тип печати

Этим термином мы обозначаем тип интервалов (или шагов) между символами (по-английски spacing). Таких типов два — monospace и proportional space. Первый обозначает, что шаг между символами — постоянный, как в машинописных текстах, второй — что шаг переменный, как в типографских текстах. Это разли-



чие имеет важное практическое следствие. Для типа monospace всегда известно знакоместо любого символа, и поэтому сравнительно легко отделить соседние знаки друг от друга, даже если между ними нет явного пробела. В случае же proportional space ситуация многократно усложняется, так как чисто геометрические соображения уже не работают, а разделение соприкасающихся символов возможно лишь на основе результатов распознавания. Лишь для очень небольшого числа программ декларирована такая возможность, но на практике они ее подтверждают далеко не во всех случаях.

Некоторые выводы

Современные читающие системы, например такие, как Omni Page, обладают целым рядом превосходных характеристик:
- высокая надежность распознавания на многих текстах;
- широкий диапазон распознаваемых шрифтов;
- автоматическая селекция текстов в документах сложного формата;
- обработка текстов пропорционального шага печати;
- наличие большого числа выходных форматов;
- обработка текста на нескольких языках;
- контекстная обработка распознанного текста;
- работа с многими типами сканеров.

И, разумеется, всегда есть удобный пользовательский интерфейс. Но при этом оказывается, что в реальных условиях, когда качество печати — среднее или невысокое, когда имеются разрывы, заплывы, соединения знаков, посторонние включения в тексте, перекос страницы, надежность распознавания резко падает, и пользователь вынужден отказываться от автоматического ввода. Таким образом, на сегодня прогресс технологии автоматического чтения больше сказался на сопутствующих показателях, чем на самом важном — надежности чтения. И наверное, не случайно, читая многочисленные обзоры, содержащие сравнительные характеристики современных OCR-систем, часто наталкиваешься на заключительную фразу приблизительно одного и того же смысла: "В конечном итоге оказалось, что этот материал лучше подготовить оператору вручную".

Таким образом, потенциальные возможности технологии автоматического чтения на сегодня далеко не исчерпаны, и следует ждать разработок, отвечающих новым, значительно более серьезным требованиям.

Особенности применения читающих систем в отечественных условиях

Попытки внедрения зарубежных читающих программ в отечественных условиях в целом оказались безуспешными. Представления первых пользователей о том, что если программа обучаемая, то ее можно обучить всему, что ей предложат, оказались несостоятельными. И здесь они столкнулись с тем, чем мы заключили предыдущий раздел: как только качество текста падает, программа становится неэффективной. Необучаемые программы не имеют сегодня надежного "кириллизованного" варианта, но хорошо представляя качество отечественного текста, в особенности машинописного, трудно ожидать хорошего результата и в ближайшем будущем. Кроме того, зарубежные программы не имеют контекстной поддержки для русского языка. Все эти причины побудили целый ряд разработчиков создать собственные варианты распознающих программ, позволивших более успешно решать задачу автоматического ввода русского текста. Многие из этих программ демонстрируют такие современные черты, как обработка многих шрифтов, разметка документа на зоны, контекстная обработка, поддержка нескольких типов сканеров и т.д.

Так что же дальше?

Несомненно, что за последние несколько лет в технологии OCR достигнут значительный прогресс. Однако остается еще больше потенциальных клиентов, нуждающихся в автоматическом вводе документов, чьим потребностям современные системы не отвечают. На наш взгляд, предстоит сделать ряд последовательных шагов для совершенствования этой технологии. Отдельные усовершенствования здесь не помогут, более того, прогресс только в средствах автоматического чтения тоже даст ограниченный результат. Необходимо создавать инфраструктуру для обработки текста с использованием OCR-систем как своего существенного элемента. Это означает, что совершенствуя свои разработки, создатели читающих систем должны не ждать пользователя, а открывать для него все новые сферы применения, о которых он еще не догадывается.

Суммируя наши впечатления, попробуем сформулировать главные черты новых читающих систем, о которых можно говорить как о системах нового поколения.

1. Существенно повышенная надежность. Этот комплексный вопрос может быть успешно решен только на

пути тщательного балансирования этапов обработки информации и установления обратных связей между ними.

- Обработка документов с целью повышения его качества. В современных системах этот этап достаточно примитивен, и мощностей даже новых персональных компьютеров может быть недостаточно для такой работы, если речь идет о текстах невысокого качества. Такая обработка может решить и свою некоторую минимальную задачу, а именно — настройка сканера на контраст документа без участия человека.
- Совершенствование алгоритмов распознавания. Эта работа должна идти по пути создания более помехозащищенных алгоритмов описания знаков, многоуровневых процедур принятия решения с более жесткой проверкой гипотез. Результаты распознавания могут оказывать влияние на предыдущие этапы по обработке изображения и описанию знаков.
- Увеличение роли контекстной обработки. Контекстному процессору передается широкий список гипотез, подлежащих проверке, но сами гипотезы выбираются только из жесткого списка близких символов.

2. Обработка документов различных форматов. Разнообразие форматов касается расположения текста на листе, размеров символов, используемых шрифтов и т.д. Данные могут быть представлены в табличном виде, на типовых бланках, быть развернутыми на 90 градусов, разделены вертикальными или горизонтальными линиями. Система должна содержать средства настройки на любые особенности такого рода.

3. Чтение сложных текстов.

Мы выделим две проблемы: чтение рукописного текста и чтение формульных данных.

Чтение рукописного текста — достаточно традиционная задача. Однако ограничения на стилизацию письма еще очень велики, и, даже не говоря о действительно свободном стиле написания, предстоит достаточно много сделать для существенного уменьшения таких ограничений.

Обработка формульных данных является актуальной задачей для чтения научной и технической информации. Авторам на сегодня не известны какие-либо системы, успешно решающие эту задачу.

4. Многоязыковость.

Многоязыковость — это современная тенденция, и лучшие из современных систем уже обладают такой возможностью, но по большей части они ограничиваются текстами на основе латинского алфавита. По нашему опыту, одной из актуальных задач является обработка текстов на языках народов, населяющих территорию бывшего СССР (включая и государства Балтии). Наибольшая трудность здесь в разработке соответствующих программ проверки правописания.

5. Развитый, но простой в использовании интерфейс.

Чтобы удовлетворить широкому спектру требований, интерфейс должен обладать возможностями программирования большого числа параметров и значительным набором команд. Естественно, простому пользователю, не программисту, манипулировать большим количеством представленных ему возможностей очень тяжело. Однако путем предварительного формирования заданий на выполнение отдельных комплексных работ можно чрезвычайно упростить взаимодействие с системой, сведя его, образно говоря, к выбору необходимой кнопки и ее нажатию.

Другая характеристика интерфейсной оболочки состоит в том, что она позволяет включать путем программирования все большее число новых функций, образуя тем самым расширяемую интегрированную среду для работы. В качестве таких дополнительных функций можно предложить графический редактор, языковый переводчик и др.

Декларируя развитие читающих систем в указанных направлениях, мы постарались значительную их часть реализовать в своей новой версии AUTOR 3.0, разрабатываемой в настоящее время. Ее наиболее важными чертами будут значительно более высокая надежность чтения, в том числе и для слипшихся знаков, обработка широкого спектра входных форматов, в частности таблиц и типовых форм, автоматическая и интерактивная разметка страницы, использование средств формирования задания на комплексную работу, многоязыковость, возможность использования как автоматических, так и ручных сканеров, и ряд других. В эту версию мы стараемся включить многие из предложений, поступа-

ющих от наших сегодняшних пользователей, которые порождены не умозрительными схемами, а реальными потребностями.

Области применения

Мы уже отмечали, что успешность применения читающих систем зависит не только от их собственного качества, но и от правильного использования, от всей инфраструктуры обработки текста. Каждая из областей применения имеет свой собственный акцент и делает ударение на различных характеристиках системы.

Издательские системы

На сегодня это наиболее очевидная сфера применения технологии автоматического чтения. Для нее характерен массовый ввод достаточно однородного материала, подготовленного как на машинках, так и типографским способом. Наиболее важными показателями системы являются надежность и скорость ввода, совместимость по выходному формату со средствами редактирования и верстки. Работа с различными типами шрифтов является обязательной, но сама смена типов происходит не очень часто.

Информационные системы

На наш взгляд, эта область активно соперничает с издательской сферой и скоро превзойдет ее по массовости применения. Огромное количество данных, пассивным грузом хранящихся на бумагах, выключено из процесса компьютерной обработки. Надежд на ручной ввод всего этого количества — практически никаких. Помимо новых данных, значительная часть которых обязательно регистрируется на машинных носителях, существуют колоссальные архивы, к которым никто и никогда не подступался, и здесь альтернатив OCR-технологии просто нет. С другой стороны, создаются запоминающие устройства огромной емкости, а средств заполнить их информацией зачастую нет. Наиболее важными чертами читающих систем для этого применения должны быть повышенная надежность распознавания, поддержанная в ряде случаев специальными средствами обработки изображений, возможность разметки документов, широкий диапазон шрифтов, многоязыковость.

Системы массового ввода данных

Примером такого рода обработки является ввод данных платежных документов, отчетов, запросов, заявок, расписаний и тому подобного. Чаще всего данные расположены в виде таблиц или типовых бланков, и они в изобилии имеются в банках, коммерческих структурах, биржах, статистических органах. И здесь бумажные документы как универсальный способ хранения информации, легко подготавливаемые любым печатным или рукописным способом, могут объединить самые разнообразные потоки информации и даже сыграть некоторую организующую роль, пока электронные средства связи охватывают лишь ограниченный круг документов.

Наиболее характерными чертами здесь являются сложность расположения данных на документах, многообразие форматов исходного текста и размеров документов.

Сфера управления

Эта сфера достаточно разнородна, в нее мы включаем обработку документов из различных трудно ограничиваемых источников информации, в том числе тех, которые могут просто "оказаться под рукой". Руководитель или коммерсант, просматривая многочисленные журнальные, газетные или иные сообщения, никогда не решится вручную вводить большое число заинтересовавших его данных в свой компьютер. Однако, имея под рукой автоматический или даже ручной сканер, универсальную и надежную OCR-систему, он с удовольствием введет нужный ему материал в собственную базу данных. И здесь, на наш взгляд, следует говорить не только о технологии автоматического ввода, а о разработке интегрированной среды, включающей помимо OCR-системы специальную базу данных, легко позволяющую включать введенный текст в соответствующий ему по содержанию раздел. Мы думаем, что многие пользователи высоко бы оценили такой интегрированный продукт, дающий им возможность оперативно и независимо собирать информацию, которая с этого момента становится доступным компонентом базы данных.

Авторы этой статьи относятся к числу тех немногих разработчиков OCR-систем, которые прошли весь путь от первых исследований по моделированию распознающих систем в 60-х годах до создания реальных современных коммерческих программ.

Наш опыт показывает, что создание читающих систем, которые сделают бумажный документ почти таким же доступным для компьютера, как дисковый файл, не пустая фантазия, а реальная перспектива ближайших 2-3 лет. Многие черты таких систем уже реализуются в новой версии программы AUTOR v.3.0.

В заключение хотелось бы сказать, что мы, авторы этой статьи, далеко не единственные авторы наших разработок. На протяжении многих лет в работе над ними активно участвовали М.А.Назаров, Ю.Л.Зарубин, М.Ю.Шкловер. Мы хотели бы поблагодарить И.А.Овсевича, В.К.Раева, и П.А.Александрова, без поддержки которых нам не удалось бы провести нашу разработку.

И.Зенкин, А.Кучеров,
Б.Мазо, А.Петров

Контактные телефоны:
(095) 287-14-63, 287-16-62, 238-20-76, 237-36-58

Что было в прошлом году на компьютерном рынке?

Ответ на этот вопрос дала аналитическая фирма DataQuest в выпущенном ежегодном исследовании мирового компьютерного рынка. IBM с трудом удалось удержать первую позицию по размеру сегмента рынка в борьбе с Apple.

Как сообщается, рынок начал вновь оживляться во второй половине 1992 года и вырос (в терминах доходов предприятий) на 7.4 процента. В целом, мировой рынок персональных машин вырос с 43.4 миллиарда долларов в 1991 году до 46.5 миллиардов в 1992-м.

"Рынок сильно улучшился в последнем квартале, что связано с ускорением перехода пользователей на мощные 486-е машины. Эту миграцию стимулировало появление Windows 3.1, Apple System 7.1 и новых программ, требующих больших компьютерных ресурсов",— заявил Брэд Смит, ведущий аналитик фирмы DataQuest.

IBM и Apple вели в течение всего года тяжкую битву за долю рынка, и, хотя IBM удалось сохранить первое место, яблочников должна распирать гордость за содеянное. Все дело в том, что у страдающей от своих размеров IBM прибыли от продажи персональных машин сократились на 18 процентов. А у Apple — увеличились на 14.2 процента. Compaq продемонстрировал рекордный объем продаж в 1992 году и его рост на 19 процентов. NEC упал с третьего на четвертое место на рынке по сравнению с прошлым годом. Фирма Dell выбралась с 11 места (в прошлом году) на пятое, практически удвоив свои прибыли за год.

"Но чтобы преуспевать и в 1993 году, фирмам придется быть очень гибкими и вовсю заниматься инновациями. Звонкость или абсолютная неизвестность имени практически не будут влиять на решения о закупке продукции у той или иной компании..."

Рынок рабочих станций продемонстрировал весьма слабый рост — всего 4.2 процента, и это после 14.5 процентов роста в 1991 году. Это было вызвано в основном тем, что все большую долю поставляемых рабочих станций составляют недорогие и менее мощные продукты. Спад на японском рынке сильно ударил по японскому рынку рабочих станций и по их продавцам. Рынок рабочих станций увеличился с 8.7 миллиарда долларов в 1991 году до 9 миллиардов в 1992-м.

"Мы видим явное изменение в тенденциях ценообразования на рабочие станции. 1992 год был первым годом, когда пять крупнейших фирм начали продавать RISC-станции менее чем за 10 тысяч долларов. Более 60% поставок рабочих станций шестью крупнейшими продавцами в третьем квартале 1992 года были системами с ценой менее 15 000 долларов".

Первая пятерка продавцов рабочих станций по сравнению с предыдущим годом не изменилась. Sun удержал лидерство с 9.8-процентным ростом. Hewlett-Packard показал 13-процентный рост, а продажи рабочих станций IBM выросли на 21 процент. В 1992 году IBM отгрызла себе еще 3 дополнительных процента рынка рабочих станций. Доходы Silicon Graphics выросли за год на 34 процента. И только один член первой пятерки уступил место на рынке — доходы Digital Equipment Corporation уменьшились с 1 до 0.8 миллиарда долларов за год.

Доля крупных систем (mainframes) на рынке уменьшилась за год на 16 процентов, с 107 до 104.5 миллиардов долларов. Это вызвано (наряду с экономическим кризисом во всем мире) такими технологическими явлениями, как активный переход от модели большого центрального компьютера к модели "клиент-сервер", сильно снижающей цену компьютерных ресурсов для конечного пользователя. Еще одним фактором было снижение цен самими производителями в попытке удержать свои позиции на рынке

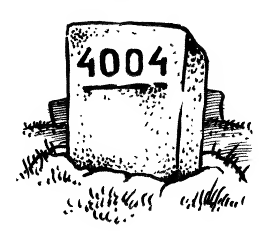
Впервые на мировом рынке миниЭВМ (mid-range computers) обогнали крупные компьютерные системы по объемам продаж.

"Но это не надо понимать как болезнь и смерть рынка больших машин. Этот рынок по-прежнему занимает примерно 20 процентов от общего числа продаваемых машин. Однако мы ожидаем, что покупатели будут и далее держаться несколько в стороне от крупных машин, предпочитая им нечто поменьше и подешевле", — сказала Нэнси Стьюарт, старший аналитик DataQuest по данному сегменту рынка.

Продажи maniframes сократились наиболее сильно — на 16 процентов по сравнению с 1991 годом, до 22.5 миллиардов долларов. Хотя IBM и сохранила свое первенство на этом участке рынка, их объем продаж уменьшился на 21 процент.

К.Чащин

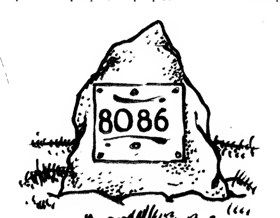
По материалам:
Newsbytes News Network,
January 5, 1993



Сердцем любого персонального компьютера является микропроцессор, причем в подавляющем большинстве случаев — это микропроцессор фирмы Intel. Эта небольшая статья поможет начинающим пользователям персональных компьютеров уяснить основные черты микропроцессоров семейства x86.

Микропроцессоры Intel: из прошлого в будущее

Первый микропроцессор — i4004 был изготовлен фирмой Intel еще в 1971 году. Первоначально он предназначался для использования в карманном калькуляторе, однако впоследствии применялся и в ряде других областей, например для систем управления светофорами. Первый микропроцессор представлял из себя 4-разрядное параллельное вычислительное устройство, выполненное по р-канальной МОП-технологии (p-MOS). Стоит, пожалуй, напомнить, что максимальная тактовая частота этого прапрадедушки современных “числодробилок“ составляла не более 750 кГц.



Реализация ряда следующих проектов фирмы Intel по разработке однокристальных микропроцессоров (i4040, i8008) возвестила о наступлении эры персональных компьютеров. Наиболее успешным был, пожалуй, проект разработки микропроцессора i8080. Кстати, впоследствии именно на этом микропроцессоре был основан компьютер “Альтаир“, для которого молодой Билл Гейтс написал свой первый интерпретатор Бейсика. Этот 8-разрядный микропроцессор был выполнен по n-канальной МОП-технологии (n-MOS), а его тактовая частота не превышала 2 МГц. Не будет преувеличением сказать, что классическая архитектура i8080 оказала огромное влияние на дальнейшее развитие однокристальных микропроцессоров. Несмотря на заслуженный успех i8080, настоящим промышленным стандартом для персональных компьютеров стал другой микропроцессор фирмы Intel.

8088

Микропроцессор i8088 был анонсирован Intel в 1979 году, а в 1981-м "Голубой Гигант" (фирма IBM) выбрал этот микропроцессор для своего первого персонального компьютера и, надо сказать, не ошибся. Одним из существенных достоинств микропроцессора i8088 была возможность (благодаря 20 адресным линиям) физически адресовать область памяти в 1 Мбайт. Здесь следует, правда, отметить, что для IBM PC в этом пространстве программам было отведено всего лишь 640 Кбайт. Хотя с внешними периферийными устройствами (дисками, видео) i8088 был связан через свою внешнюю 8-разрядную шину данных, его внутренняя структура (адресуемые регистры) позволяла работать с 16-разрядными словами. Как известно, на системной шине IBM PC для передачи данных отведено 8 линий (1 байт). Первоначально микропроцессор i8088 работал на частоте 4,77 МГц, однако впоследствии были разработаны его клоны, рассчитанные на более высокую тактовую частоту (например, 8 МГц).

8086

Этот чип, появившийся в 1978 году, стал популярен в основном благодаря компьютеру Compaq DeskPro. Программная модель (доступные регистры) этого микропроцессора полностью совпадает с моделью i8088. Основное отличие этих микропроцессоров состоит в различной разрядности внешней шины данных: 8 разрядов у i8088 и 16 разрядов у i8086. Понятно, что более высокой производительности с новым микропроцессором можно было достичь только при использовании компьютера, на системной шине которого под данные предусмотрено 16 линий. Адресная шина микропроцессора i8086 по-прежнему позволяет адресовать не более 1 Мбайта памяти.

80286

Опираясь на архитектуру i8086 и учитывая запросы рынка, в 1982 году фирма Intel выпустила свой новый микропроцессор i80286. Надо сказать, что этот чип появился практически одновременно с новым компьютером фирмы IBM — PC/AT. Наряду с увеличившейся производительностью, этот микропроцессор (i80286) отличало то, что он мог теперь работать в двух режимах — реальном и защищенном. Если первый режим был (за рядом исключений) вполне похож на обычный режим работы i8088/86, то второй использовал уже несколько изощренную технику управления памятью. В частности, защищенный режим работы позволял, например, таким программным продуктам, как Windows 3.0 и OS/2, работать с оперативной памятью выше 1 Мбайта. Благодаря 16 разрядам данных на шине ISA, которая была впервые использована в IBM PC/AT286, микропроцессор i80286 мог обмениваться с периферийными устройствами 2-байтными сообщениями. 24 адресных линии нового микропроцессора позволяли в защищенном режиме обращаться к 16 Мбайтам памяти. В микропроцессоре i80286 впервые на уровне микросхем были реализованы мультизадачность и управление виртуальной памятью.

80386DX, или оригинальный 80386

В 1985 году фирмой Intel был анонсирован первый 32-разрядный микропроцессор i80386. Первым компьютером, использующим этот микропроцессор, был Compaq 386. Полностью 32-разрядная архитектура (32-разрядные регистры и 32-разрядная внешняя шина данных) в новом микропроцессоре была дополнена встроенным устройством управления памятью MMU (Memory Management Unit). Это устройство позволяло легко переставлять сегменты из одного места памяти в другое (свопинг) и освобождать драгоценные килобайты стандартной (640 Кбайт) памяти. Напомним, что 32 адресные линии микропроцессора позволяют физически адресовать 4 Гбайта памяти. С появлением таких программных утилит, как 386MAX фирмы Qualitas и QEMM фирмы Quarterdeck Office Systems, стало возможным размещать резидентные (TSR) программы и драйверы устройств в верхние адреса памяти. Несмотря на полную 32-разрядную архитектуру, для обеспечения совместимости с существующими устройствами, с периферийным оборудованием микропроцессор i80386 может обмениваться 8- и 16-разрядными словами. Первоначальная версия микропроцессора работала на частоте 16 МГц, в настоящее время имеются версии для частот 20, 25 и 33 МГц.

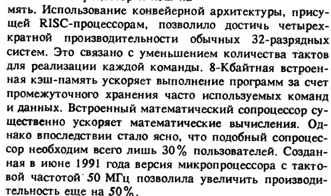
80386SX

Более дешевая альтернатива 32-разрядного процессора i80386DX появилась только в 1988 году. Это был (достаточно популярный сейчас и в нашей стране) микропроцессор i80386SX. В отличие от своего старшего "собрата", новый микропроцессор использовал 16-разрядную внешнюю шину данных. Это было особенно удобно для стандартных PC/AT, системная шина которых (ISA) использует, как известно, только 16 линий данных. Благодаря дешевизне нового изделия многие

производители "железа" стали заменять теперь уже устаревший микропроцессор i80286 на более производительный i80386SX. Одним из решающих факторов для замены была полная совместимость 32-разрядных микропроцессоров: программное обеспечение, написанное для i80386DX (например, Paradox 386 или 386MAX), корректно работало и на i80386SX. Надо отметить, что уже к концу 1988 года микропроцессор i80386SX выпускался в количествах, существенно превосходящих рекордные показатели для i80386DX.

80486DX, или оригинальный 80486

На осенней выставке Comdex в 1989 году фирма Intel впервые анонсировала микропроцессор, который содержал более миллиона транзисторов (а точнее, 1,2 миллиона) на одном кристалле и был полностью совместим с процессорами ряда x86. Напомним, что на кристалле первого члена этого семейства — микропроцессора i8088 насчитывалось только около 29 тысяч транзисторов. Новая микросхема впервые объединила на одном чипе такие устройства, как центральный процессор, математический сопроцессор и кэш-память. Использование конвейерной архитектуры, присущей RISC-процессорам, позволило достичь четырехкратной производительности обычных 32-разрядных систем. Это связано с уменьшением количества тактов для реализации каждой команды. 8-Кбайтная встроенная кэш-память ускоряет выполнение программ за счет промежуточного хранения часто используемых команд и данных. Встроенный математический сопроцессор существенно ускоряет математические вычисления. Однако впоследствии стало ясно, что подобный сопроцессор необходим всего лишь 30% пользователей. Созданная в июне 1991 года версия микропроцессора с тактовой частотой 50 МГц позволила увеличить производительность еще на 50%.



80486SX

Появление нового микропроцессора i80486SX фирмы Intel вполне можно считать одним из важнейших событий 1991 года. Уже предварительные испытания показали, что компьютеры на базе i80486SX с тактовой частотой 20 МГц работают быстрее (примерно на 40%) компьютеров, основанных на i80386DX с тактовой частотой 33 МГц. Микропроцессор i80486SX подобно оригинальному i80486DX содержит на кристалле кэш-контроллер и кэш-память, а вот математический сопроцессор у него заблокирован. Значительная экономия (благодаря исключению затрат на тестирование сопроцессора) позволила фирме Intel существенно снизить цены на новый микропроцессор. Надо сказать, что если микропроцессор i80486DX был ориентирован на применение в сетевых серверах и рабочих станциях, то i80486SX послужил отправной точкой для создания мощных настольных компьютеров. Вообще говоря, в семействе микропроцессоров i486 предусматривается несколько новых возможностей для построения мультипроцессорных систем: новые команды поддерживают механизм семафоров памяти, аппаратно реализованное выявление недостоверности строки кэш-памяти обеспечивает согласованность между несколькими модулями кэш-памяти и т.д. Для микропроцессоров семейства i486 допускается адресация физической памяти размером 4 Гбайта и виртуальной памяти размером 64 Тбайта.

80386SL

К концу 1991 года 32-разрядные микропроцессоры стали стандартными для компьютеров типа лэптоп и ноутбук, однако такие обычные микросхемы, как i386DX/SX, не полностью отвечали требованиям разработчиков портативных компьютеров. Для удовлетворения потребностей этого сегмента рынка фирмой Intel и был разработан микропроцессор i80386SL. Данный микропроцессор представляет собой интегрированный вариант микропроцессора i80386SX, базовая архитектура которого дополнена несколькими дополнительными контроллерами. По существу, все компоненты, необходимые для построения совместимого с ISA-шиной портативного компьютера, сосредоточены в двух микросхемах: микропроцессоре i80386SL и периферийном контроллере i82360SL. В набор i386SL впервые введено новое прерывание, называемое System Management Interrupt (SMI), которое может быть использовано для обработки событий, связанных, например, с управлением потребляемой мощности. Вместе с математическим сопроцессором i80387SL данный набор микросхем позволяет создать 32-разрядный компьютер на площади, лишь немного превышающей размер игральной карты.

80486SL

Микросхема i486SL представляет из себя самый производительный в настоящее время процессор серии SL, разработанный фирмой Intel. Анонсированная в конце 1992 года, эта микросхема объединяет характерные черты двух представителей процессорных семейств Intel: i486DX и i386SL. По производительности новый

процессор не уступает i486DX, но благодаря пониженному напряжению питания (3,3 В) и развитой технологии управления энергопотреблением (как в i386SL) он может эффективно использоваться в портативных компьютерах. Производительность систем на базе i486SL может существенно улучшиться благодаря 16-разрядной шине высокоскоростного периферийного интерфейса PI, которая поддерживает быстрый интерфейс графического дисплея и устройства хранения информации на основе флэш-памяти. По некоторым оценкам, системная плата компьютера на базе i80486SL будет на 60% меньше, чем при использовании i386SL, а среднее время автономной работы компьютера-блокнота (около 3 часов) может увеличиться примерно на один час только за счет использования нового микропроцессора.

80486DX2

В марте 1992 года фирма Intel объявила о создании второго поколения микропроцессоров i486. Новые микропроцессоры, названные i80486DX2, обеспечили новую технологию, при которой скорость работы внутренних блоков микропроцессора в два раза выше скорости остальной части системы. Тем самым появилась возможность объединения высокой производительности микропроцессора с внутренней тактовой частотой 50(66) МГц и эффективной по стоимости 25/33-мегагерцевой системой. Новые микросхемы по-прежнему включают в себя центральный процессор, математический сопроцессор и кэш-память на 8 Кбайт. Компьютеры, построенные на базе микропроцессоров i486DX2, работают приблизительно на 70% производительнее тех, что основаны на микропроцессорах i486DX первого поколения. Следует напомнить, что технология удвоения частоты применяется также в процессорах OverDrive, которые призваны поддерживать производительность систем, основанных на i80486DX/SX, на уровне требований новых программных средств.

Pentium

Новый микропроцессор фирмы Intel, называемый Pentium, должен быть анонсирован в начале 1993 года. Название нового микропроцессора является зарегистрированной торговой маркой корпорации Intel, таким образом, в системах "Intel Inside" микропроцессор 586 фигурировать не будет. Системы, построенные на базе Pentium, полностью совместимы со 100 миллионами персональных компьютеров, использующих микропроцессоры i8088, i80286, i80386, i80486. Новая микросхема выполнена по BiCMOS 0,8-микронной технологии и содержит более 3,3 миллиона транзисторов. Суперскалярная архитектура позволяет достичь производительности около 100 MIPS.

А.Борзенко



Компьютерные вирусы-92: от плохого к худшему, или Есть ли вирусы на Марсе?

На горе лежит дискета,
У нее запорчен бут.
Через дырочку в конверте
Ее вирусы грызут.

Тяжела и неказиста
Жизнь простого программиста.
Народный фольклор

Замысел этой статьи родился на традиционной (как оказалось) конференции по системному программированию, проходившей в Абрау-Дюрсо. Собиравшаяся по утрам публика, в ожидании когда нагреются камни на осеннем пляже, развлекала себя докладами, сообщениями и дискуссиями на программистские и околопрограммистские (т.е. наиболее злободневные) темы. Пользователи ЕС-10хх, СМ, PDP-11, VAX, IBM PC рассуждали о программировании реляционных баз данных, спеллчекеров, параллельных систем, мультимедиа и о многом другом. Удалось и нам (Дмитрию Мостовому, Юрию Павловичу Лященко и мне) сказать свое слово в защиту темы компьютерных вирусов, которая, честно говоря, мало волнует пользователей не "MS-DOS на IBM PC".

На мой взгляд, тема вирусов стоит несколько отдельно от всех остальных программистских задач. Практически все проблемы, решаемые при помощи вычислительной техники, являются продолжением целенаправленной борьбы человека с окружающей его природой. Природа ставит человеку длинное нелинейное дифференциальное уравнение в трехмерном пространстве — человек набивает компьютер процессорами, памятью, обвешивает пыльными проводами, много курит и в итоге решает это уравнение (или пребывает в состоянии уверенности, что он его решил). Природа дает человеку кусок провода с вполне определенными характеристиками — человек придумывает алгоритмы передачи как можно большего объема информации по этому проводу, терзает его модуляциями, сжимает байты в биты и терпеливо ждет сверхпроводимости при комнатной температуре. Природа (в лице фирмы IBM) дает человеку 640 Кбайт+12 МГц — и человек не спит ночами, опять много курит*, оптимизируя коды очередной базы данных. И так далее.

А вот борьба с компьютерными вирусами является борьбой человека с человеческим же разумом (тоже в некотором смысле проявлением природных сил, хотя на этот счет имеется более одного мнения). Это — борьба умов, поскольку задачи, стоящие перед вирусологами, ставят тоже люди. Они придумывают новый вирус — а нам с ним разбираться. Затем они придумывают вирус, в котором разобраться очень тяжело — но мы и с ним разбираемся. И сейчас, наверняка, сидит где-то за компьютером парень, который не глупее меня, и мучается над очередным монстром, в котором мне придется разбираться целую неделю, а потом еще одну неделю отлаживать алгоритм "лечилки" против этого вируса.

Но это все околофилософские вопросы, а уже давно пора переходить к конкретным. Итак...

Где водятся вирусы? И как долго они будут нас беспокоить?

Основная питательная среда для массового распространения вируса в ЭВМ, на мой взгляд, состоит из следующих необходимых компонентов:

* Сам автор статьи не курит. (Прим.редакции.)

- незащищенность операционной системы (ОС);
- наличие разнообразной и довольно полной документации по операционной системе и "железу";
- широкое распространение этой ОС и этого "железа".

Если в операционной системе присутствуют элементы защиты информации, как это сделано практически во всех ОС, вирусу будет крайне трудно поразить объекты своего нападения, так как для этого потребуется (как минимум) взломать систему паролей и привилегий. В результате работа, необходимая для написания вируса, окажется по силам только профессионалам высокого уровня (вирус Морриса для VAX — пример этому). А среди профессионалов, на мой взгляд, уровень порядочности все-таки немного выше, чем у потребителей их продукции, и, следовательно, число созданных и запущенных в большую жизнь вирусов сокращается.

Еще для массового производства вирусов необходимо и достаточное количество информации о среде их обитания. Какой процент от общего числа системных программистов, работающих на мини-ЭВМ в операционках UNIX, VMS и т.д., знает систему управления процессами в оперативной памяти, полные форматы выполняемых файлов и загрузочных записей на диске (то есть информацию, крайне необходимую для создания вируса)? И, следовательно, какой процент от их числа в состоянии вырастить настоящего полноценного зверя?

Ну а по поводу широкого распространения ОС как необходимого условия для вирусного нашествия и говорить надоело: на 1000 программистов приходится 100, способных написать вирус, на эту сотню приходится один, который эту идею претворит в жизнь. Теперь полученную пропорцию умножаем на число тысяч программистов — и получаем результат: 1500 или даже 2000 полностью IBM-совместимых вирусов. Десятки (или сотни?) вирусов для Apple Macintosh.

Вот и получается, что вирус в IBM PC — явление не случайное, а подчиненное неким законам и правилам, а, следовательно, ответ на вопрос, вынесенный в заголовок, будет следующим: если на Марсе есть MS-DOS на IBM PC, то там обязаны присутствовать и вирусы.

Для того чтобы прикинуть продолжительность нашествия компьютерных вирусов, надо оценить время одновременного сосуществования приведенных выше необходимых условий.

Довольно очевидно, что в обозримом будущем фирмы IBM и Apple не собираются уступать массовый рынок своим конкурентам (на радость Apple- и IBM-программистам), даже если для этого им придется объединить усилия. Не представляется возможным и усечение потока информации по наиболее распространенным системам, так как это ударит по числу приложений для них, а, следовательно, и по их продаваемости.

Остается одно: защита ОС. Когда это произойдет? Когда на столы секретарш, бухгалтеров, брокеров и т.д., и т.п. придут ОС с разграничением доступа к ресурсам системы? Вопрос для меня сложный, и пусть он прогнозируется не мною. Допустим, на это потребуется *5* лет (для стран отдельно построенного социализма — плюс еще *5* лет). Стало быть, время вирусного нашествия ограничивается пятью или десятью годами. Остается только ждать.

История компьютерных вирусов: от древности до наших дней

Мнений по поводу того, когда появился первый компьютерный вирус, очень много. Мне доподлинно известно только одно: на машине Бебиджа его не было, а на IBM-360/370 уже был (вирус "Christmas tree").

На этом разговор о вымерших ископаемых предлагаю считать завершенным. Поговорим о новейшей истории: от "Венского", "Падающих букв" и далее. Те, кто начал работать на IBM PC аж *5* или *8* лет назад, еще не забыли повальную эпидемию этих вирусов. Буквы сыпались по экранам, а толпы пользователей неслись к специалистам по ремонту дисплеев (сейчас все наоборот: винчестер сдох от старости, а валят на неизвестный передовой науке вирус). Затем компьютер заиграл чужеземный гимн "Yankee Doodle", но чинить динамики уже никто не бросился — очень быстро разобрались, что это — вирус, да не один, а целый десяток.

Так вирусы начали заражать файлы. Скачущий по экрану шарик ознаменовал победу вируса и над Boot-сектором (см. эпиграф). Все это очень не нравилось пользователям IBM PC — и появились противоядия. Первым попавшимся мне антивирусом был Anti-Kot: это легендарный Олег Котик выпустил в свет первые версии своей программы, которая уничтожала целых 4 (четыре!) вируса. Кстати, всем, кто до сих пор сохранил копию этого антивируса, предлагаю немедленно ее стереть (да простит меня Олег Котик!) как программу вредную и ничего, кроме траты лишних нервов и ненужных телефонных звонков, не приносящую. К сожалению, Anti-Kot определяет "Иерусалимский" вирус по комбинации "MsDos" в конце файла, а какой-то другой антивирус эти самые буквы аккуратно прицепляет ко всем файлам с расширением COM или EXE.

Время шло, вирусы плодились. Все они были чем-то похожи друг на друга, лезли в память, цеплялись к файлам и секторам, периодически грохали файлы, дискеты и винчестеры. Одним из первых откровений стал вирус "V-4096" — первый из известных мне файловых вирусов-невидимок. Этот вирус перехватывал int 21h и, при обращении через DOS к зараженным файлам, изменял информацию таким образом, что файл появлялся перед пользователем в незараженном виде. Но это была только надстройка вируса над MS-DOS. Не прошло и года, как электронные тараканы полезли внутрь DOS (вирус-невидимка "V-512"). Идея невидимости продолжала приносить свои плоды и далее: летом 1991 года пронесся, кося компьютеры как бубонная чума, вирус "Driver-1024". "Да-а-а!"— сказали все, кто в нем копался.

Но бороться с невидимками было довольно просто: почистил RAM — и будь спокоен, ищи гада и лечи его на здоровье. Побольше хлопот доставляли самошиф-

рующиеся вирусы. Ведь для их идентификации и удаления приходилось писать специальные подпрограммы, отлаживать их. Но на это никто тогда не обращал внимания, пока... Пока не появились вирусы нового поколения, те, что на Западе называют polymorphic-вирусы. Эти вирусы используют другой подход к невидимости: они шифруются (в большинстве случаев), а в расшифровщике используют команды, которые могут не повторяться при заражении различных файлов. Простейшим примером этого является следующий расшифровщик (пока еще не polymorphic!):

```
    mov reg_1, count         ; reg_1, reg_2, reg_3
  mov reg_2,.key           ; выбираются из
  mov reg_3, _offset       ; AX,BX,CX,DX,SI,DI,BP
                           ; count, key, _offset
                           ; также могут меняться
_loop:
  xxx byte ptr [reg_3], reg_2 ; xor, add или sub
  dec reg_1
  jxx _loop                ; ja или jnc
  ; а дальше следуют зашифрованные коды и данные
```

Похожие алгоритмы используются в вирусах “Word“, “Phoenix“ и многих других. В результате НИ ОДИН байт кода самого вируса и его расшифровщика не является постоянным при заражении различных файлов. Но и это не составляет большой проблемы, так как остаются постоянные биты, по которым и можно зацепиться за вирус и начать его расшифровку (можно, конечно, расшифровывать все файлы подряд, даже те, которые вируса не содержат, но это отразится на скорости работы антивирусной программы).

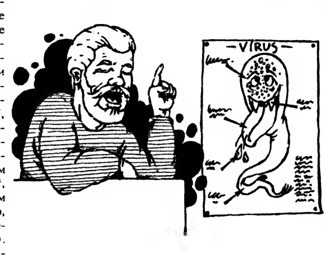
Следующий шаг компьютерных мичуринцев был в селекции мутантов, в которых нет ни одного постоянного БИТА информации. Это было достигнуто довольно легко: приведенные выше инструкции (или их эквиваленты) переставлялись местами от заражения к заражению, разбавлялись ничего не меняющими командами типа NOP, STI, CLI, STC, CLC, DEC неиспользуемый регистр, XCHG неиспользуемые регистры и т.д. В результате в начале файла, зараженного подобным вирусом, идет набор бессмысленных, на первый взгляд, инструкций, причем некоторые комбинации, которые вполне работоспособны, не берутся фирменными дизассемблерами (например, сочетание CS:CS: или CS:NOP). И среди этой “каши“ из команд и данных изредка проскальзывают MOV, XOR, LOOP, JMP — инструкции, которые действительно являются “рабочими“. К подобным вирусам можно отнести “Starship“, “Amoeba“, “Mutant“, причем последний вирус, видимо, отечественного происхождения (умеем, если захотим!). Но и с этими уродами справиться не составило труда. Это было достигнуто двумя методами (или их комбинацией):

1) в каждом конкретном вирусе алгоритм шифровки не менялся, поэтому можно извлечь ключ шифровки прямо из шифрованного участка в файле, если это не получается, значит файл не заражен;
2) сканирование начала файла: отбрасывание мусора и поиск действительно “рабочих“ команд.

В результате скорость обнаружения этих polymorphic-вирусов совсем незначительно отразилась на общем быстродействии антивирусов. Но дальше — больше. Наступил год 1992-й. Он был ознаменован новой победой селекционеров: на свободу выполз вирус “Pogue“ (другое название — “MtE“). Это животное использует чрезвычайно сложный polymorphic-алгоритм, в результате работы которого в расшифровщике вируса могут встретиться операции SUB, ADD, XOR, ROR, ROL в произвольном количестве и порядке. Загрузка и изменение ключей и других параметров шифровки производятся также произвольным набором операций, в котором могут встретиться более половины инструкций процессора 8086 (ADD, SUB, TEST, XOR, OR, SHR, SHL, ROR, MOV, XCHG, JNZ, PUSH, POP...) со всеми возможными режимами адресации. И самое смешное в том, что эта штука работает, причем работает без ошибок!

Вот так. Задача обнаружения вируса значительно усложнилась. Как теперь отличить зараженный файл от незараженного? Ведь и в зараженных, и в незараженных файлах могут встретиться самые разнообразные инструкции в самом разнообразном порядке. Не говорю уже о том, что надо бы вытащить из расшифровщика алгоритм шифровки, его ключи и расшифровать тело вируса.

Решал я эту проблему около недели, копался в листингах, залез в сотню зараженных файлов. В результате за 3 дня был написан эмулятор (довольно примитивный) процессора 8086, который эмулирует (не трассирует!) работу файла, и, если файл заражен вирусом, выдает его (вируса) расшифрованное тело. В большинстве случаев незараженные файлы отметаются эмулятором сходу. Но если файл заражен или “похож“ на зараженный, то антивирус “приседает“ в среднем на 1 секунду на каждый “подозрительный“ файл (справедливо для AT-286/16). Умножим эту цифру на число проверяемых на винчестере файлов... Итого: вместо 1 минуты на 32-Мбайтный винчестер в скором будущем потребуется минут 10. Что будет, когда таких вирусов появится два, три,... а потом много? Если честно, то сейчас их уже 4, причем один из



них (вирус "Bomber") использует несколько большее число команд при выработке своего начала. Так что скоро могут появиться таблички: "С 9:00 до 13:00 компьютер на антивирусной профилактике".

Плохо? Плохо. Но все вышесказанное — мышиная возня по сравнению с дальнейшим развитием событий: вирус "Pogue" был расхакан, затем из него был извлечен алгоритм генерации расшифровщиков (MtE-алгоритм), этот алгоритм был откомпилирован в OBJ-файл, снабжен необходимой документацией (довольно полной) и помещен на BBS. БИ-БИ-ЭС! Вы не ослышались! Лежит теперь этот аккуратненький ZIP, в котором исходники, OBJ-файлы, примеры и подробные объяснения, как вызывать из любого вируса MtE-алгоритм, как линковаться и т.д. И сейчас любой козел может превратить самый безобидный вирус в жуткого мутанта! И не исключено, что, если не будут разработаны алгоритмы быстрого поиска MtE, время сканирования винчестера будет измеряться ЧАСАМИ, а не минутами! Таблички сменятся на: "С 1-го по 15-е число ежемесячно компьютер находится на профилактике".

Вот так. Тенденция развития событий — от плохого к наихудшему. Что ожидать дальше? Что готовит нам год 1993-й???

Тенденции развития компьютерной ассенизации

Жизнь принесла десяток типов антивирусных программ (фаги, полифаги, детекторы, сторожа, иммунизаторы и т.д.) и несколько сотен антивирусных разработок. На мой взгляд, практически все они потеряли актуальность: сторожа (или блокировщики) из-за того, что появилось много вирусов, которые либо встраиваются в DOS, и их вызовы невозможно отличить от "родного" вызова DOS, либо используются принципиально новые идеи, например, заражение файлов только при их копировании. Иммунизаторы же бессильны против вирусов-невидимок, к тому же они могут испортить иммунизируемые файлы, и т.д.

Теряют актуальность и наиболее популярные антивирусы — полифаги, то есть программы, обнаруживающие и удаляющие массу вирусов, конкретных и известных автору (авторам) данного антивируса. Да-да! Полифаги становятся неактуальными! В том числе и старые версии программы -V.EXE.

При современном положении дел, когда число новых вирусов измеряется десятками (а иногда и сотней) экземпляров в месяц, подобные полифаги не могут обеспечить надежной защиты. Просто не хватает времени, чтобы подключать все новинки в антивирусную базу. И довольно часто большое число вирусов стоит в очереди на обработку, а зараженные пользователи либо ждут и мучаются, либо ищут другие способы избавления от заразы (например, путем форматирования всего, что форматируется). Тем более смешна ситуация, когда пользователи антивируса прошлого года изготовления живут в уверенности абсолютной защиты их любимых "писишек" от злобных пришельцев. Как бы не так, господа! Версию антивируса следует менять как можно чаще, а антивирусы, изготовленные в прошлом году, следует пустить под нож ради экономии места на диске.

Чем же следует пользоваться?

Остались программы проверки целостности файлов и секторов на дисках. Наиболее симпатичная для меня разработка в этой области — система ADINF Дмитрия Мостового. Она показалась мне настолько интересной, что часть ее функций я решил вставить в новую версию -V.EXE.

А что же делать, если программа ADINF кричит, что все файлы увеличились в размере на 2 Кбайта? Вот тут следует применять полифаги второго поколения, разработкой одного из которых моя команда занималась более 5 месяцев. Отличительная особенность этой системы состоит в том, что база данных, по которой антивирус определяет и удаляет вирусы, является ОТКРЫТОЙ. То есть квалифицированный пользователь (если он, конечно, во-первых, в состоянии самостоятельно разобраться в алгоритме работы вируса, а, во-вторых, приобретет редактор базы) может быстро и самостоятельно написать обнаруживалку и лечилку на новый внезапно появившийся вирус. Тщательно протестировав свое изделие (то есть добавку к основной антивирусной базе), он может положить новую базу на BBS и этим избавить многих других пользователей от многочасовых мучений. В базу данных заложен алгоритм поиска вирусов в файлах и около 20 стандартных методов удаления вирусов (в том числе метод DELETE). Если же метод удаления вируса нестандартный, то оказавшийся неподалеку системщик может написать на языке Си или Ассемблере собственную программу для обнаружения и удаления вируса, откомпилировать ее в OBJ-файл и поместить в базу данных. При старте антивируса этот OBJ будет считан из базы и автоматически слинкован с основным EXE-модулем. Хочу добавить, что база данных, поставляемая с антивирусным комплексом, зашифрована, что практически исключает ее модификацию злоумышленником.

Этим в значительной степени снижается нагрузка на разработчиков антивирусных программ. И теперь больше времени можно потратить на борьбу с новой чумой — вирусами типа polymorphic и другими зверями, которые окажутся не по зубам простому честному пользователю.

Е.Касперский
E-mail: eugene@kami.npimsu.msk.su
Тел.: (095) 499-15-00



Ваш покорный слуга имел неосторожность посетить крупнейшую американскую компьютерную выставку — Comdex Fall, проходившую с 15 по 20 ноября 1992 года в городе Лас-Вегасе, на Невадщине. Дальнейшее — его путевые заметки. Автор благодарит П.Антонову, Д.Бланкенхорна, Х.Брайнскомба, В.Вудс, Л.Рорбах, С.Устинова, П.Цукера и Newsbytes News Network за помощь в создании данного материала.

Репортаж с Комдекса, или Выставка в пустыне

На выставке было представлено около полутора тысяч фирм, вокруг стендов которых можно ходить километров примерно шестьдесят, что и делали примерно 150 тысяч посетителей. Руководители промышленности выступали с публичными заявлениями, показывали массу интересных продуктов (но ничего существенно нового). Рекламные проспекты и информационные издания раздавались десятками и сотнями килограммов.

Очередь на регистрацию участников в первый день работы выставки растянулась на 2-3 километра.

Основные идеи очевидны — компьютеры становятся все мощнее и все меньше. Мощнее настолько, что не хватает программистов для разработки программ, и меньше настолько, что машину можно засунуть в любой карман и продолжать быть подключенным к глобальным информационным сетям. Это уже реальность... Однако обо всем по порядку.

Из речей руководителей промышленности на очередном пленуме

Начнем с Билла Гейтса. Самый богатый человек США, президент и председатель фирмы Microsoft, посвятил свою речь тому, куда идет программная промышленность и как она будет развиваться в ближайшее время.

Для начала он вспомнил таксиста, который вез его от аэропорта до гостиницы. Тот приставал к президенту с вопросами о проблемах в Windows 3.1. "Но чего я не понимаю, так это для чего таксист использует Окна??" — удивлялся Билл.

Более очевиден для него факт, что все изменяется очень быстро. Не только Microsoft поставляет в месяц миллион копий Windows, но более миллиона PC будет продано в 1993 году с Windows for Workgroups и сетевыми платами — втыкай и работай. Windows можно будет запускать на машинах, различающихся по цене в 50 раз. Ожидается встраивание Окон в школьные доски и телефоны-автоматы (несколькими такими телефонами с клавиатурами и экранами я пользовался лично).

Нетерпение прессы по поводу медленного развития компьютеров с рукописным вводом Гейтс умерил. "Надо быть терпеливыми, — сказал он. — Я честно верю в эту технологию".

Гейтс указал также на изменение парадигмы продажи и установки компьютеров. "Во времена IBM'овских гигантов в цену машины входили установка и поддерж-

ка. Сейчас же есть большая необходимость в фирмах, которые могли бы давать грамотные советы о том, как включить компьютер и как его обслуживать, ибо большинство производителей этого не делают".

Гейтс привел также пример банковской программной системы, которую крупный банк проектировал для своих внутренних нужд. Разработчики написали прототип системы на Visual Basic и только потом поняли, что дальше ничего переделывать не надо — все прекрасно работает и так. А создано все было всего за несколько недель.

Потом Билл начал рассказывать и показывать новую версию технологии OLE (object linking and embedding). OLE 2.0 будет сохранять объект в содержащем его документе, но менять структуру меню на ту программу, которая этот объект создавала. То есть выбирая для редактирования фотографию, уже вставленную в текстовый документ, вы получите на экране органы управления графического пакета, которым эта фотография была отсканирована, но документ вместе с текстом останется на экране без изменений.

Билл хотел было продолжить свой рассказ, но время прямой спутниковой трансляции истекало, и ему пришлось закруглиться на середине выступления. О том, что было дальше, мир так и не узнал...

Другое выступление было более веселым.

Филипп Кан, президент фирмы Borland, сказал в программном выступлении, что аппаратное обеспечение меняется так быстро, что программистам, дабы не отстать от поезда, нужно СУЩЕСТВЕННО перестраивать способы своей работы.

Продемонстрировав, как объектно-ориентированный подход к программированию помог их местному полицейскому департаменту в разработке требуемых информационных систем, Кан заметил, что если не перейти на объектно-ориентированные технологии, написать программу раньше того, как она успеет физически и морально устареть, будет невозможно.

Кроме того, как гласит "закон Филиппа", производительность труда программиста в команде из N человек уменьшается в кубический корень из N раз, то есть в команде из 8 человек каждый из них будет работать вдвое хуже.

Объектное программирование по методу "разделяй и властвуй" и только оно сможет дать возможность написания требуемых программ, которые к 2000 году, в среднем, будут иметь 10 миллионов строк кода.

В живой беседе с Гордоном Муром, президентом фирмы Intel, директора сошлись на том, что процессор Pentium 2000 фирмы Intel будет иметь порядка 100 миллионов транзисторов, а компьютеры — сотни мегабайт ОЗУ и гигабайт внешней памяти. Гров считает, что к 2000 году 1 (один) мегабайт памяти будет стоить 1 (один) доллар...

"Наш бизнес фундаментально меняется, программная промышленность станет более похожа на полупроводниковую, с заводами, производящими различные компоненты — либо готовые текстовые процессоры, либо маленькие кусочки для них..."— отметил Гров.

Гейтс также поддержал эту идею, говоря, что огромные программы сейчас слабо умеют взаимодействовать друг с другом. Именно эта область, по его словам, будет активно разрабатываться фирмами в ближайшее время.

Главному в мире продавцу средств для разработки объектно-ориентированных программ все это представляется так: одни будут собирать компоненты, другие монтировать готовые программные изделия из этих компонентов, третьи — продавать творения первых двух. Объектно-ориентированный подход даст вычислительные возможности в руки каждому. Если персональные компьютеры изменили способ труда людей — объектно-ориентированный подход изменит самих людей. "И в этом наше будущее", — сказал Гейтс.

В кулуарах Comdex Fall'93

До и после выступления г-н Кан летал на приличной высоте над выставочными павильонами на небольшом самолетике с дли-и-инным плакатом Borland Flies Here (Тут Летает Борланд).

Желающие увидеть это светило бизнеса занимали очередь за 2 часа, что оказалось несколько меньше очереди на Билла Гейтса. Очередь из огромного количества респектабельных белых людей в костюмах и галстуках с чемоданчиками в руках равномерно заполняла весь первый этаж большой гостиницы Хилтон. Число желающих услышать эти два выступления существенно превышало пропускную способность выделенных для выступления помещений.

Гейтс, или Гигадоллар Билл, был главной достопримечательностью Комдекса. Легенды о нем продолжают расти и множиться. Так же как Сэм Волтон — его предшественник на должности самого богатого человека Америки, Гейтс не испорчен деньгами и славой.

Например, на съемки для обложки журнала Time он приехал в грязной потертой фирменной майке, и ему пришлось в прямом смысле снимать более чистую с кого-то из своих сотрудников. Говорят, что Билл регулярно питается гамбургерами и молоком во время ночных бдений со своими программистами, продолжающихся до трех утра.

На книжные прилавки Америки в декабре прошлого года легла книга "Гейтс", рекламирующая себя следующим образом: "Билл Гейтс имеет достаточно денег, чтобы купить каждому американскому мужчине, женщине и ребенку по экземпляру этой книги. Но он, скорее всего, не сделает этого". Это скандальная книжка.

Самый богатый человек Америки прилетел на Комдекс на самом дешевом самолете в салоне самого дешевого — экономического — класса. Когда об этом рассказывали представителям трудового люда в Лас-Вегасе — таксистам и работникам казино — они смеялись и всякими другими способами выражали восхищение. Не удивился только Марк Эппли, президент фирмы Traveling Software, который тоже отличается немалым состоянием. "Я летел тем же рейсом", — сказал он...

Пока программисты провозглашают новые подходы, все становится мобильным

Мотороловский президент Джордж Фишер считает, что в самом недалеком будущем можно будет увидеть беспроводные версии всего представленного на выставке — разнообразные комбинации голоса, данных и картинок. Достичь этого позволят быстро прогрессирующее слияние радио, компьютеров и электроники и стремительное развитие каждой из этих отраслей в отдельности.

"С цифровой обработкой сигналов и распознаванием голоса, — уверен г-н Фишер, — ваши клавиатуры в скором времени вам будут совершенно не нужны. Нейросети уже сейчас могут имитировать работу человеческого мозга. Так вот, все это начинает работать, помещаясь на вашей ладошке. Окончательной целью является общество, представляющее собой взаимодействующую глобальную сеть. И это не столь дикая мечта сейчас".

А пока этого не произошло, 30 миллионов человек в мире уже пользуются беспроводными сотовыми телефонами и еще примерно столько же — односторонними системами радиопоиска...

Пока мотороловский президент рассуждал о перспективах, AT&T демонстрировала эти перспективы. Творение называлось Hobbit. Эта машинка размером с маленькую книжку, умещающаяся в небольшом кармане, будет продаваться за 2-3 тысячи долларов.

Устройство поддерживает рукописный ввод информации, электронную почту, факс и беспроводную связь со всем миром.

Сердце устройства — собственно чип Hobbit, производства AT&T. Вставлять его в разнообразные маленькие устройства будут Toshiba, NEC и ряд других производителей. Примерно в течение года чип будут вставлять и испытывать перед тем, как начать активно продавать.

Специалисты фирмы считают, что "персональным коммуникатором" будут пользоваться даже те, кому сейчас компьютер вообще не нужен. Но они объективно оценивают способности рынка. По их мнению, "Hobbit должен стоить менее 500 долларов, чтобы стать массовым товаром".

А если у вас нет Хоббита, то можно удовлетвориться и более скромными palmtop-компьютерами, к которым можно подключать интересные внешние устройства, соответствующие стандарту PCMCIA. (Правда, совсем никто не знает, как это правильно произносится.)

Год назад группа фирм выпустила вторую версию спецификации PCMCIA, стандарт на размер устройства и соединительную шину для подключения его к компьютеру. И разнообразные устройства уже сейчас производятся и продаются более чем 130 фирмами — карты ОЗУ и ПЗУ, модемы, сетевые адаптеры, винчестеры, сканеры, карты Token Ring, эмуляторы 3270, карты ввода изображений и т.п.

Примерно половина из последователей стандарта решили показывать свои изделия под одной крышей на выставке. Очень интересное зрелище. Как вам идея миниатюрного компьютера, в который вставлен модем размером 2x5x0,5 см и проводочком присоединен к маленькому сотовому телефону, то есть к телефонной трубке с антенной? Хочешь разговаривать — разговаривай, хочешь электронную почту почитать — пожалуйста.

А еще на этой выставке были представлены такие обязательные атрибуты компьютерной промышленности, как фильтры для мониторов, столы для компьютеров и программы бухгалтерского учета. Как и везде, раздавались бесплатные шарики, пакетики и значки. Выставка как выставка, ничего особенного...

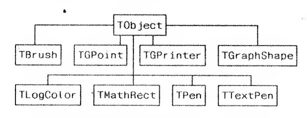
К.Чащин



Там, где кончается OWL, — начинается OGL...

Реализованная фирмой Borland объектно-ориентированная библиотека ObjectWindows (для компиляторов C++ и Pascal) является оболочкой вокруг вызовов функций Windows API, позволяя тем самым создавать объектно-ориентированные Windows-программы. В то время как библиотека ObjectWindows содержит объекты для реализации пользовательского интерфейса и приложения в целом и ряд объектов общего назначения, библиотека Object Graphics фирмы Whitewater Group (в настоящее время — части фирмы Symantec) содержит объекты для работы с модулем GDI (Graphics Device Interface).

Как и в OWL, в основе иерархии объектов находится объект TObject (из библиотеки OWL). Также из библиотеки OWL используются объекты TCollection, TScroller, TStream, TDosStream, TBufStream, TWindowsObject, TDialog и TWindow. Отметим, что изначально обе библиотеки были разработаны для языка Actor, а затем для языков Pascal и C++. Таким образом, используя библиотеки OWL и OGL, мы получаем объектно-ориентированный интерфейс для создания практически любых Windows-программ.



Объекты—наследники объекта TObject

В этом обзоре мы рассмотрим организацию объектно-ориентированной библиотеки Object Graphics. Читатели, желающие более подробно ознакомиться с функциями модуля GDI, могут прочитать о них в КомпьютерПресс №№ 11,12'92.

Организация библиотеки

Как мы отметили выше, ряд объектов OGL являются наследниками объектов библиотеки OWL. Эти объекты представлены на рисунке.

Два объекта являются наследниками объекта TStream: TGStream и TMemoryStream. Наследники объекта TCollection — объекты TPalette и TPointColl. От библиотеки OWL наследуется объект TScroller — TGScroller.

Панели диалога для настройки и управления принтером реализованы на основе объекта TWindowsObject через объект TDialog: TAbortDlg и TSetupDlg.

Все объекты, реализующие графические примитивы, являются наследниками объекта TShape.

Для всех окон, отображающих графические примитивы, существует объект TGWindow (наследник объекта TWindow из OWL). Поле Picture этого объекта содержит указатель на объект TPicture, который является коллекцией графических примитивов. Отображение содержимого окна происходит с помощью метода GPaint.

Обычно объекты добавляются в окно один за одним с помощью метода Add. Например:

```
......
Picture^.Add(Text);
Picture^.Add(Rectangle);
......
```

После добавления объекта в список этот объект автоматически перерисовывается.

Основное окно приложения реализуется (по аналогии с OWL) как объект—наследник объекта TGWindow. Конструктор такого окна может выглядеть следующим образом:

```
Constructor TMyDemoWindow.Init(AParent: PWindowsObject; ATitle: PChar);
Begin
  TGWindow.Init(AParent, ATitle);
  ...
  {Инициализация для объекта TMyDemoWindow}
End;
```

Метод TGWindow.GPaint вызывается при обработке сообщения wmPaint и управляет отображением содержимого окна.

Простейшее приложение, использующее библиотеку OGL, может выглядеть следующим образом:

```
{-----------------------------------------------------------------------------------------

     Программа OGL_MIN: Простейший пример использования библиотеки ObjectGraphics Library.
          Компилятор: Turbo Pascal for Windows   Алексей Федоров/КомпьютерПресс.
-----------------------------------------------------------------------------------------}

Program OGL_MIN;

Uses OGL1, OGL2, OGL3, WObjects;

Type

 { Объект-наследник объекта TApplication }
  TMinApp = Object(TApplication)
    Procedure InitMainWindow;                  Virtual;
  End;

 { Объект-наследник объекта TGWindow }
  PMinWindow = ^TMinWindow;
  TMinWindow = Object(TGWindow)
    Constructor Init(AParent: PWindowsObject);
  End;

Constructor TMinWindow.Init(AParent: PWindowsObject);
Var
  ALabel: PLabel;
Begin
  TGWindow.Init(AParent, 'OGL DEMO');
  Attr.X := 10;  Attr.Y := 10;
  Attr.W := 600;  Attr.H := 300;
  New(ALabel, Init(0, 0, 'OGL Demo', tc_Tools));
  ALabel^.TextPen^.SetFont(fs_Script);
  ALabel^.TextPen^.SetStyle([ts_Italic]);
  ALabel^.TextPen^.SetHeight(100);
  Picture^.Add(ALabel);
End;

Procedure TMinApp.InitMainWindow;
Begin
  MainWindow := New(PMinWindow, Init(nil));
End;

Var
  MinApp: TMinApp;

Begin
  MinApp.Init('OGL_MIN_APP');
  MinApp.Run;
  MinApp.Done;
End.
```

В приведенном примере показано создание окна, в котором отображается строка ´OGL Demo´, созданная с помощью объекта TLabel. После инициализации этого объекта изменяются его характеристики: шрифт — Script, стиль — наклонный, высота — 100. Затем объект TLabel вставляется в определяемую полем Picture коллекцию графических объектов, принадлежащих окну.

Основные объекты

Программа, написанная с использованием OGL, может отображать графические примитивы на экране, принтере или добавлять изображения к объекту типа TBitMap. В последнем случае объект TBitMap выступает в качестве объекта — контейнера, содержащего другие объекты.

Объекты, реализованные в этой библиотеке, можно разделить на две категории: управляющие объекты и графические объекты. Объекты TPen и TBrush реализуют соответствующие элементы Windows (они являются наследниками объекта TObject), тогда как объект TTextPen объединяет такие атрибуты, как шрифт, размер, цвет, стиль, и используется для отображения текста. Объект TPen, описывающий средство отображения "карандаш", содержит поля, задающие его основные характеристики: цвет, стиль, режим отображения и ширину.

Объект TGraphSpace управляет координатной системой, преобразованием осей, увеличением (zoom) изображения и ассоциируется с графическим устройством (экран, принтер или растровое изображение).

Ядро Windows (модуль GDI) содержит большое число функций для отображения различных графических примитивов: линий, прямоугольников, эллипсов и т.п. Все эти примитивы реализованы в виде отдельных объектов. Библиотека OGL содержит такие объекты, как TEllipse — для отображения эллипсов и окружностей, TRectangle — для отображения прямоугольников, TRoundRect — для отображения прямоугольников с закругленными углами, TLine — для отображения линий, TPolygon — для отображения многогранников, TPolyLine — для отображения ломаных линий и TPie — для отображения секторов круговых диаграмм.

Такие объекты содержат поля, описывающие их основные характеристики, устанавливаемые при вызове конструктора. Изменение значений этих полей может происходить следующим образом:

```
Line^.Pen^.SetCombo()

    или

Line^.Brush^.SetPattern()
```

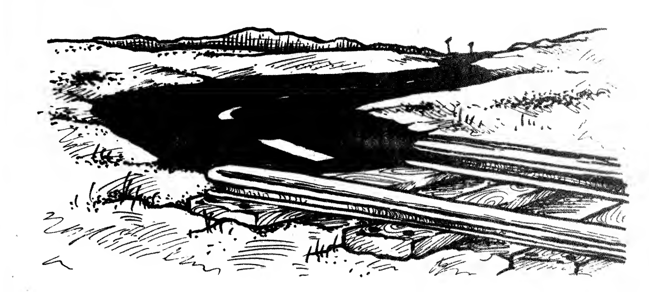
Помимо объектов, реализующих графические примитивы, в библиотеке находятся и более комплексные объекты. Например, объект TBezier используется для отображения кривых Безье, объект TPolyMark — для отображения границ вокруг объекта, TRichText — это коллекция текстовых строк с различными шрифтами, стилями и цветами. В библиотеке реализован также объект TCurve, на основе которого можно создавать объекты для отображения сложных фигур (один из таких объектов — объект TBezier).

Одним из наиболее важных объектов в OGL является объект TPicture. Этот объект представляет собой коллекцию других графических объектов. Каждый объект типа TGWindow имеет поле Picture типа PPicture. Поле Picture может быть использовано для поиска объекта: реализованы методы для работы с коллекциями ти-

па At, Find, FindWithID и т.п. После того как указатель на объект получен, над ним можно выполнять любые действия — изменять характеристики, перемещать и т.п.

Работа с растровыми изображениями

Для работы с растровыми изображениями очень удобен объект TGBitmap. Его методы позволяют сохранять растровое изображение на диске и загружать его с диска. Этот объект может использоваться также как "выходное устройство". Например, загрузка растрового изображения с диска выглядит следующим образом:



```
Var
 Bitmap : PGBitmap;

{Обычно при описании окна}
Bitmap := New(PGBitmap,Init(0,0,0,0,tc_NoTools);
Picture^.Add(Bitmap)
```

```
........
{Загрузить из файла}
If Bitmap^.LoadBMPFile('DEMO.BMP', Space) <> 0
 Then {Ошибка}
  Begin
   MessageBox()
  End
  Else {Успешно}
  Begin

  { Работа с растровым изображением }

  End
```

Объект TGBitmap содержит поля, позволяющие определить все характеристики растрового изображения: размер: BitsPixel, PixHeight, PixWidth, тип DIBInfo, палитру и т.п. Также поле Handle содержит ссылку на растровое изображение, что делает манипуляции с ним более простыми.

Перемещение объектов

Три метода объекта TGWindow предназначены для реализации перемещения графических объектов. Метод BeginDrag вызывается при нажатии левой кнопки манипулятора "мышь". Метод Drag вызывается при перемещении манипулятора с нажатой левой кнопкой, а метод EndDrag — когда левая кнопка мыши отпущена. Для реализации перемещения объектов необходимо переопределить эти три метода в объекте — наследнике объекта TGWindow. Реализация этих процедур может выглядеть следующим образом:

```
Procedure TMyGWindow.BeginDrag;
Begin
  If SomeGObject^.Contains(MousePt) Then
   Begin
    SetGCursor(cs_Mover);

    {
      Действия по подготовке перемещения |
      обычно | настройка порта
    }

   End;
End,
```

```
Procedure TMyGWindow.Drag;
Begin

{
  Перемещение объекта в позицию MousePt
  Перерисовка, обычно через SomeGObject^.FastDraw
}

End;

Procedure TMyGWindow.EndDrag;
Begin

{
  Перемещение объекта в позицию MousePt
  Перерисовка
  Восстановление курсора: SetGCursor(cs_Normal);
}

End;
```

Сохранение файлов — формат OGL

В библиотеке определен собственный формат для хранения графических изображений. Для работы с файлами используется объект TStream (объект TGStream, наследник объекта TBufStream) библиотеки OWL. Файл, имеющий расширение .OGL, состоит из заголовка и объектов TGraphSpace и TPicture. Любая программа, создаваемая с использованием OGL, может использовать эти файлы. Например, программа ObjectDraw, входящая в комплект поставки, позволяет сохранять созданные изображения в формате .OGL и считывать их для редактирования.

Документация и примеры

Документация состоит из руководства и справочной системы в формате Windows Help. К сожалению, содержимое справочной системы не отражено в документации. Поэтому при работе с библиотекой OGL необходимо постоянное использование справочной системы.

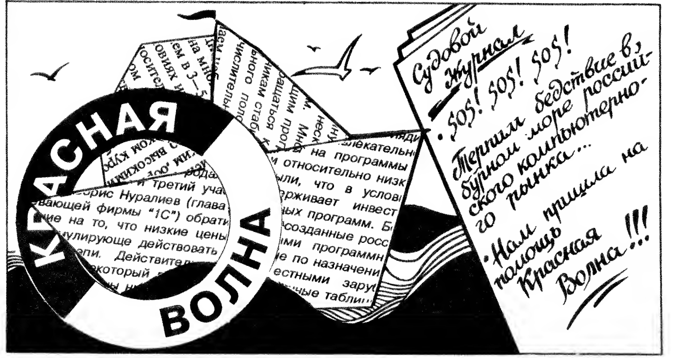
В комплекте с библиотекой поставляется ряд примеров, показывающих основные приемы работы с OGL:

GHELLO.PAS.	Простейшее OGL-приложение
BMPSCROL.PAS.	Пример загрузки объекта TGBitmap, отображения его в окне и скроллинга растрового изображения в окне
GCONTROL.PAS.	Пример создания элемента управления
GMDIDEMO.PAS.	Пример создания многооконного интерфейса, каждое окно которого имеет собственную иконку
MOVERECT.PAS.	Пример реализации перемещаемого графического объекта
OGLSCROL.PAS.	Пример загрузки файла в формате OGL
PICTURE.PAS.	Пример работы с объектом TPicture

Кроме того, поставляется исходный текст демонстрационной программы ObjectDraw, являющейся наиболее комплексным примером использования библиотеки OGL. Программа ObjectDraw — это графический редактор со всеми необходимыми атрибутами, возможностью выбора различных средств отображения, изменения их свойств, загрузки и сохранения графических файлов, а также вывода их на устройство печати.

Библиотека Object Graphics представляет собой хороший инструмент для программистов, работающих в среде Windows, и позволяет создавать программы, манипулирующие графическими примитивами значительно проще при использовании технологии объектно-ориентированного программирования.

А. Федоров



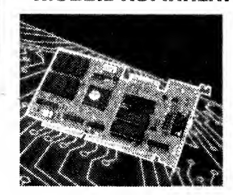
КРАСНАЯ ВОЛНА - ЭТО:

превращение рублей в доллары, закупка, транспорт, таможня, биржевые операции

И КОМПЬЮТЕРЫ ВЫСШЕГО КАЧЕСТВА Тел. (095) 320-43-00, Fax (095) 320-33-44

CLSIC LTD

ЛЮБЫЕ КОМПЛЕКТУЮЩИЕ ДЛЯ ВАШИХ КОМПЬЮТЕРОВ



Внешнеторговая фирма "ЭЛСИ"

предлагает со склада в Москве :

СИСТЕМНЫЕ ПЛАТЫ, ВИНЧЕСТЕРЫ,

ДИСКОВОДЫ, МОНИТОРЫ,

КЛАВИАТУРЫ, КОРПУСА,

ПРИНТЕРЫ, МОДЕМЫ,

ПЛАТЫ ARCNET и ETHERNET

Контрактные поставки из США и Юго-Восточной Азии. Факс 480-2238

MOSCOW
LENINSKY pr., 35-a
tel. 952-0218, 952-0238
fax. 976-5280, 480-2238



МОСКВА
ЛЕНИНСКИЙ ПР., 35-а
тел. 952-0218, 952-0238
факс.976-5280, 480-2238



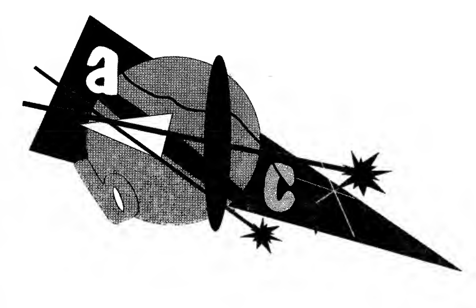
Мы продолжаем публикации в нашей новой рубрике рассказом о ноутбуке модели Encyclopedia, который был предоставлен редакции Компьютер-Пресс фирмой НИТА. Это уже второй компьютер-блокнот, предлагаемый этой фирмой на отечественном рынке. Разумеется, представлением только ноутбуков мы ограничиваться не будем. Так что следите за нашими публикациями в рубрике "Впечатления".

Компьютер Encyclopedia

Первые впечатления

На первый взгляд ноутбук модели Encyclopedia никаких особых эмоций не вызывает: ноутбук как ноутбук. Но, как известно, первое впечатление часто обманчиво. В данном случае это было действительно так. Представленная модель портативного компьютера имеет, например, встроенный адаптер для сетевого электропитания. И это при весе около 6,6 фунта (3 кг). Не следует забывать, что данная цифра включает и вес встроенного привода флоппи-дисков размером 3,5 дюйма. Кроме того, мы впервые, пожалуй, рассказываем о компьютере-блокноте, оснащенном встроенным вентилятором. Включается он автоматически только при питании ноутбука от сети, так как понятно, что при работе от аккумуляторов это было бы непозволительной роскошью.

Максимальные габаритные размеры компьютера составляют 9,2 на 11,5 на 2,1 дюйма (235x293x52 см). Данная модель оснащена достаточно быстрым винчестером на 60 Мбайт и опера-



тивной памятью размером 2 Мбайта. Базовый микропроцессор i386SX может функционировать на двух тактовых частотах (8 и 21 МГц). Все интерфейсные разъемы (за исключением разъема для внешней клавиатуры) расположены на задней панели корпуса ноутбука. Сетевой выключатель достаточно удобно размещен в плоскости клавиатуры. С правой нижней стороны корпуса легко доступен сменный блок расширения, включающий дополнительный никель-кадмиевый аккумулятор и портативный факс-модем.

Хотелось бы особо отметить развитую программу Setup BIOS (Standard и Advanced), которая помимо стандартных (дата, время, тип диска и т.п.) позволяет вводить некоторые дополнительные интересные установки и параметры. Например, имеется возможность выбора последовательности загрузочных дисководов не A: C:, как обычно, а C: A:. Можно также выбрать опцию быстрого переключения адресной линии A20, что особенно удобно при работе с сетевыми операционными системами. Информация пользователя на винчестере может быть легко защищена, если установить в Advanced Setup опцию пароля.

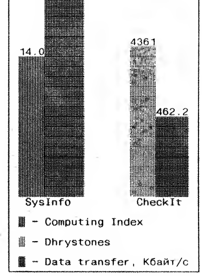
Кстати, данная модель компьютера-блокнота оснащена "чипсом" BIOS фирмы AMI (American Megatrends), в который, как обычно, включена полная программная диагностика микропроцессора и периферийных устройств. Можно отметить такие полезные утилиты для винчестера, как выбор оптимального коэффициента чередования (interleave), анализ поверхности и низкоуровневое форматирование.

Несмотря на все перечисленные достоинства, не обошлось и без недостатков. Например, технология экономии энергопотребления данного компьютера-блокнота оставляет, вообще говоря, желать лучшего. Поскольку об этом речь еще впереди, заметим только, что рассматриваемую модель ноутбука можно, пожалуй, рекомендовать для работы в личном автомобиле (справа или слева от шофера). Об этом, в частности, свидетельствует специальный кабель питания от "прикуривателя", входящий в комплект данного компьютера.

Вместе с ноутбуком Encyclopedia поставляются сетевой кабель, кабель для питания от автомобильного аккумулятора, переходной кабель для подключения клавиатуры, кабель для подключения факс-модема, документация на ноутбук, дискета с утилитами, документация и дискеты коммуникационных программных пакетов, а также отличная сумка для переноски. Все вместе это размещается в небольшой удобной коробке.

Клавиатура

Модель Encyclopedia имеет 83-клавишную клавиатуру, обычную, кстати, для многих ноутбуков. Лишь небольшая часть клавиш эмулируется нажатием служебной



клавиши Fn и клавиш, на которых нанесены дополнительные символы, выделенные синим цветом. Это, в частности, функциональные клавиши F11 и F12 (эмулируемые соответственно через F1+Fn и F2+Fn), а также клавиши "серый плюс" и "серый минус". Наличие последних клавиш, как известно, дает определенные преимущества, например, при работе в Norton Commander для отметки определенных файлов. Заметим, что в режиме встроенной цифровой клавиатуры эти клавиши также эмулируются. Для изменения тактовой частоты работы процессора необязательно вызывать программу Setup, так как это можно сделать, используя комбинации клавиш на встроенной (или внешней) клавиатуре. Тактовая частота работы процессора становится максимальной при нажатии сочетания трех клавиш Ctrl-Alt-"+" и соответственно минимальной — при нажатии Ctrl-Alt-"-".

На клавиатуре имеется по две клавиши Shift и Alt, но только одна клавиша Ctrl, что может создать определенные неудобства, если вы привыкли пользоваться программой—русификатором клавиатуры с "горячей" клавишей "правый" Ctrl. Клавиша Caps Lock расположена в нижнем ряду клавиатуры рядом с клавишей Fn. Полноразмерные клавиши управления курсором выстроены в форме "перевернутое Т", что, как известно, очень удобно. Десять функциональных клавиш (F1-F10), хотя по размеру и меньше обычных, но расположены привычным образом — в верхней части клавиатуры. Там же находятся клавиши NumLock, ScrollLock, Print Screen и Pause. Что же касается клавиш, интенсивно используемых, например, при редактировании текстов, то они расположены "столбиком" в правой части клавиатуры: Home, PgUp, PgDn, End. Режим встроенной цифровой клавиатуры (расположенной, разумеется, в поле алфавитно-цифровых клавиш) доступен после нажатия выделенной клавиши NumLock.

Дисплей

Ноутбук Encyclopedia оснащен TST VGA-дисплеем с задней подсветкой экрана. В режиме REVERSE цвет экрана практически "бумажно-белый". При разрешающей способности 640 на 480

точек такой LCD-экран может воспроизводить до 16 оттенков серого цвета. Видеоконтроллер фирмы Cirrus Logic оснащен 256 Кбайтами памяти, что позволяет поддерживать воспроизведение 17 стандартных VGA-режимов, в том числе и двух весьма экзотических extended-режимов с разрешением 800 на 600 точек. В стандартные режимы входят, разумеется, и все режимы, полностью совместимые с MDA, CGA и EGA.

На экране размером чуть больше 22 см (8,8 дюйма) в 16 оттенках серого цвета могут эмулироваться 16- или 256-цветная палитра. Загружаемый драйвер RAMBIOS может перемещать часть выполняемого кода из ПЗУ в оперативную память, что существенно ускоряет выполнение видеофункций. Другая специальная утилита (MXW) позволяет работать в достаточно редких режимах, поддерживающих 100- и 132-колоночные тексты.

Как известно, LCD-экран с подсветкой является одним из самых "прожорливых" элементов компьютера-блокнота. Однако после загрузки утилиты NBK_SW, если вы не работаете с мышью или клавиатурой в течение некоторого промежутка времени, называемого "таймаут", может происходить гашение LCD-экрана что соответственно уменьшает общее энергопотребление компьютера. При использовании данной TSR-программы без параметра она работает просто как утилита, изменяющая (при нажатии соответствующих "горячих" клавиш) режимы работы экрана: Reverse, Sleep и т.п. Значение параметра таймаута может изменяться в пределах от 0 до 60 минут.

Дисковая память

Ноутбук оснащен 2,5-дюймовым винчестером емкостью 60 Мбайт. Среднее время доступа составляет приблизительно 16 мс, а скорость передачи — около 1100 Кбайт/с, что, вообще говоря, впечатляет. Как и в подавляющем большинстве современных компьютеров-блокнотов, для экономии энергии аккумуляторов в модели Encyclopedia предусмотрено отключение винчестера, которое происходит, если к нему нет обращения в течение заданного промежутка времени. Этот промежуток (таймаут) задается через параметр специальной утилиты HDD_PWD. Данная утилита может, в частности, и вовсе отключить экономичный режим, если значение этого параметра равно 0. Понятно, что в таком случае винчестер будет работать непрерывно. Указанный режим работы имеет смысл, конечно, только при питании компьютера-блокнота от сети. Временной интервал (таймаут), по истечении которого происходит "останов" винчестера, может варьироваться в диапазоне от 1 до 16 минут. По умолчанию он составляет 5 минут.

Работа в автономном режиме

Среднее время работы от аккумулятора для модели Encyclopedia составляет не более 2 часов (и это только при активном использовании экономичных режимов). Кроме встроенного (внутреннего) никель-кадмиевого аккумулятора данный ноутбук имеет возможность установки дополнительного, который располагается на блоке расширения вместе с факс-модемом. Подзарядка аккумулятора в ускоренном режиме (при отключенном питании компьютера) выполняется в течение 2-3 часов.

Как уже отмечалось, в модели Encyclopedia технология экономии энергии аккумуляторов для работы ноутбука в автономном режиме недостаточно развита. Тем не менее уменьшить потребляемую мощность можно, например, используя программу Setup, выбрав значение тактовой частоты процессора, равное 8 МГц. Впрочем, скорость работы процессора может быть изменена и нажатием комбинации "горячих" клавиш. Для экономии мощности, потребляемой винчестером и LCD-экраном от аккумулятора, предусмотрены специальные утилиты, уже рассмотренные выше. Кстати, при загрузке утилит SWITCHER или NBK_SW существует возможность принудительного отключения подсветки экрана (через комбинации "горячих" клавиш).

Предупреждение пользователя об истощении энергии аккумуляторов (начинает светиться соответствующий индикатор) следует всего за несколько минут до их полного разряда. За 3-4 минуты начинают звучать и гудки динамика.

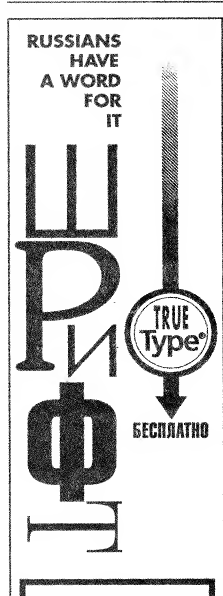
Возможности расширения

Несомненным преимуществом ноутбука является наличие сменного блока расширения, на котором в базовой конфигурации находятся факс-модем, рассчитанный как на прием, так и на передачу факсов (Send-Receive), и дополнительный аккумулятор. Системная память компьютера может быть легко расширена до 4 Мбайт путем добавления очень распространенных модулей памяти SIMM емкостью 1 Мбайт. Хотя установка таких плат обычно не слишком сложна, но в данном случае требует, как правило, вмешательства специалиста.

При работе в офисе с ноутбуком Encyclopedia удобно использовать привычную полноразмерную 101-клавишную клавиатуру и цветной CRT VGA-дисплей. Для подключения этих устройств на задней и боковой стенках корпуса компьютера имеются специальные разъемы. Мышка или трекбол без труда включаются в один из двух последовательных портов. Для работы с принтером предусмотрен интерфейс параллельного порта.

Программное обеспечение и документация

Ноутбук Encyclopedia (как это ни удивительно) поставляется без установленной на диске операционной системы. В комплект вместе с документацией включена дискета



с программами-утилитами, о которых уже шла речь. Не упоминалась ранее, пожалуй, только утилита для управления режимами работы LCD-экрана — EAGLE, которая является неотъемлемой частью прикладного "софта" для большинства компьютеров-блокнотов.

Что касается документации по ноутбуку, то можно сказать, что она рассчитана на пользователя начального уровня подготовки, поскольку практически не содержит никакой справочной информации. Основное внимание в ней уделяется аспектам именно пользовательской работы: включение, установка Setup и т.п.

Для программной поддержки штатного модуля факс-модема в комплект с компьютером включены также программные пакеты BitCom версии 5.22 и BitFax/SR версии 3.07E.

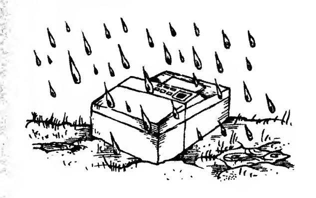
Ноутбуки модели Encyclopedia изготовляются тайваньской фирмой HONOTRON. В течение годичного гарантийного срока пользователи могут получить необходимый сервис и техническую поддержку.

На момент написания статьи стоимость этого компьютера-блокнота составляла 1350 долларов.

Все интересующие вас вопросы можно задать специалистам фирмы **НИТА** по любому из *телефонов: (095) 157-34-21, 157-10-01, 157-77-78, 157-78-41; факс (095) 157-34-21*. Вас ждут по адресу: *Москва, ул.Острякова, д.1*.

А.Борзенко





В нашей постоянной рубрике сегодня мы поговорим о том, как работают чернильные струйные принтеры (Ink Jet). Семейство подобных принтеров постоянно пополняется новыми моделями таких известных фирм, как Hewlett-Packard, Brother International, Eastman Kodak, Canon, Star Micronics и т.д. Помимо настольных струйных принтеров большим успехом пользуются и аналогичные портативные модели. О некоторых из таких устройств, предлагаемых на российском рынке, мы планируем рассказать более подробно. В этой небольшой статье изложены некоторые принципы работы струйных чернильных принтеров.

Струйные принтеры

Несмотря на то что принтеры могут казаться пользователю устройствами, на первый взгляд, достаточно прозаическими, рассказ о них может быть не только очень длинным, но и интересным. В данном случае мы не будем "растекаться мыслью по древу", а начнем, пожалуй, с классификации подобных устройств.

Как известно, все печатающие устройства можно подразделить на последовательные, строчные и страничные. Принадлежность принтера к той или иной из перечисленных групп зависит от того, формирует он на бумаге символ за символом или сразу всю строку, а то и целую страницу. Например, все без исключения лазерные принтеры можно отнести к группе страничных печатающих устройств, а вот более привычные матричные Epson'ы, Microlin'ы и т.п. — к группе последовательных.

Кстати, несколько слов по поводу матричных принтеров. Как известно, идея подобных печатающих устройств заключается в том, что все мыслимые (и немыслимые) знаки воспроизводятся ими из набора отдельных точек, наносимых на бумагу тем или иным способом. Кроме этого, не следует забывать, что все печатающие устройства (за исключением, пожалуй, страничных) могут быть "ударными" (impact) и "безударными" (non-impact). Большинство принтеров, работающих (и продаваемых) сейчас с IBM PC-совместимыми компьютерами, могут быть причислены к группе последовательных, "ударных" и матричных печатающих устройств: вертикальный ряд (или два ряда) игл вколачивает краситель с ленты прямо в бумагу, формируя последовательно символ за символом. Такое засилье "игольчатых" вполне объясняется приемлемым качеством их печати и невысокой ценой сменного красителя (ленты).

Рассматриваемые ниже струйные чернильные принтеры относятся к классу последовательных, матричных, но "безударных" печатающих устройств. Здесь следует еще раз оговориться, так как, вообще говоря, существуют также чернильные строчные и чернильные страничные принтеры, также относящиеся к классу "безударных" устройств. В этой статье они не рассматриваются.

Обычно "безударными" принтерами называются такие устройства, у которых носитель печатаемой информации не касается бумаги. Не требуется, конечно, пояснять, что "безударные" печатающие устройства работают практически бесшумно. Если продолжить уточнение признаков принадлежности печатающих устройств к отдельным группам, можно сказать, что последовательные "безударные" матричные струйные чернить-

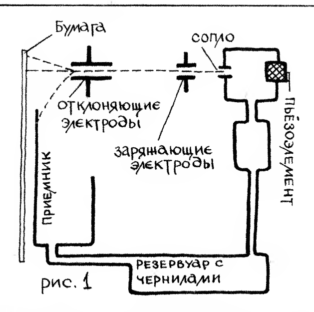


рис. 1

ные принтеры в свою очередь подразделяются на устройства непрерывного (Continuous drop, Continuous jet) и дискретного (Drop-on-demand) действия. Последние в своей работе опять же могут использовать либо "пузырьковую" технологию (Bubble-Jet), либо пьезоэффект. Подобная (правда, увы, неполная) классификация печатающих устройств полностью соответствует классификации фирмы Mannesmann Tally, хорошо известной своими принтерами.

Устройства непрерывного действия

У чернильных устройств, как впрочем и у "ударных" матричных принтеров, печатающая головка движется только в горизонтальной плоскости, а бумага подается вертикально. Сопла (канальные отверстия) на печатающей головке, через которые разбрызгиваются чернила, соответствуют "ударным" иглам. Количество сопел у разных моделей принтеров, как правило, может варьироваться от 12 до 64. Поскольку размер каждого сопла существенно меньше диаметра иглы (тоньше человеческого волоса), а количество сопел может быть больше, то получаемое изображение (теоретически) должно быть в этом случае четче. К сожалению, это не всегда так, и очень многое зависит от качества используемой бумаги (все-таки чернила!).

Схематическое изображение чернильного принтера, использующего технологию continuous drop, приведено на рис. 1. В простейшем случае принцип действия подобного устройства основан на том, что струя чернил, постоянно испускаемая из сопла печатающей головки, направляется либо на бумагу (для нанесения изображения), либо в специальный приемник, откуда чернила снова попадают в общий резервуар. В рабочую камеру чернила подаются микронасосом, а элементом, задающим их движение, является, как правило, пьезодатчик. Для того чтобы направить струю чернил в определенные места на бумаге (или в приемник), необходимы отклоняющие электроды. Разумеется, что струя чернил должна электризоваться, предварительно пройдя через заряжающие электроды. Описанный выше принцип действия печатающего устройства использует сегодня очень небольшое количество принтеров. Самыми распространенными являются более прогрессивные технологии.

Устройства дискретного действия

Итак, большинство современных чернильных принтеров используют технологию Drop-on-demand, которая основана на дискретном выпрыскивании капель чернил на бумагу из сопел печатающей головки. В настоящее время на практике для нанесения чернил на бумагу

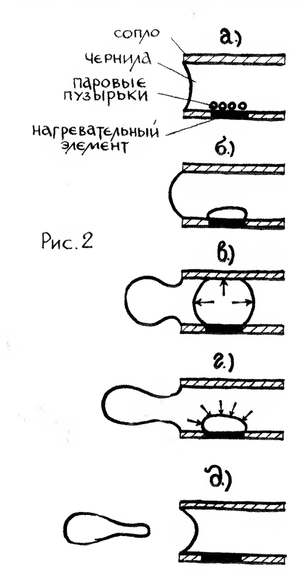


Рис. 2

наиболее широко используется два дискретных метода. Различают так называемые Bubble-Jet-принтеры и принтеры, использующие для управления соплом пьезоэлемент. Надо сказать, что при технической реализации оба этих метода практически эквивалентны, как по затратам, так и по качеству получаемого изображения.

При использовании Bubble-Jet-метода в каждом сопле находится маленький нагревательный элемент (обычно это тонкопленочный резистор). Этот элемент может находиться либо в непосредственной близости с самим соплом, либо на стороне входного канала сопла. При пропускании тока через тонкопленочный резистор последний за несколько микросекунд нагревается до температуры около 500 градусов и отдает выделяемое тепло непосредственно окружающим его чернилам. При резком нагревании образуется чернильный паровой пузырь, который старается вытолкнуть через выходное отверстие сопла необходимую порцию (каплю) жидких чернил. Поскольку при отключении тока тонкопленочный резистор также быстро остывает, паровой пузырь, уменьшаясь в размерах, "подсасывает" через входное отверстие сопла новую порцию чернил, которые занимают место "выстреленной" капли. Схематично этот процесс изображен на рис. 2.

Как уже было сказано, второй метод для управления соплом использует пьезоэлектрический элемент. Как известно, обратный пьезоэффект заключается в деформации пьезокристалла под воздействием электрического поля. Изменение размеров пьезоэлемента, расположенного сбоку выходного отверстия сопла, приводит к выбрасыванию капли и приливу через входное отверстие новой порции чернил (рис. 3).

Преимущества и недостатки

Практически все преимущества матричных принтеров автоматически можно распространить и на соответствующие струйные чернильные принтеры. Для них (теоретически) возможно создание любых типов шрифтов с различными атрибутами (с учетом, разумеется, количества сопел), не представляет особой сложности и работа в графическом режиме и т.д. Правда, как уже отмечалось, по четкости и резкости изображения некоторые модели чернильных принтеров не всегда могут поспорить с "ударными" матричными печатающими устройствами. Однако способность некоторых чернильных принтеров формировать капли разного размера, особенно при изображении наклонных линий и окружностей (для работы в графическом режиме), позволяет им создавать изображения, которые невозможно воспроизвести даже на качественном 24-игольчатом принтере.

К одному из существенных недостатков описываемых устройств относится, к сожалению, их невысокая скорость печати. Физические ограничения по скорости очевидны — в основном задержка определяется временем наполнения чернилами пустого пространства в

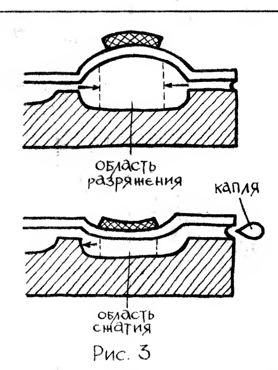
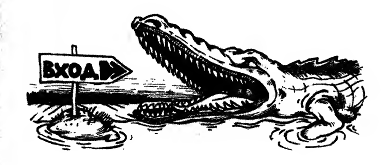


Рис. 3

сопле печатающей головки. За редким исключением, скорость чернильных струйных принтеров для высококачественных текстов (letter quality) не превосходит 100 cps (char per second).

Тем не менее, к очевидным достоинствам технологии струйных принтеров относится, например, возможность создания достаточно дешевых и качественных цветных печатающих устройств. Лучшим тому подтверждением могут служить новые модели цветных принтеров DeskJet 500C и DeskJet 550C фирмы Hewlett-Packard. Но это тема уже другого разговора.

А.Борзенко



Локальная сеть и шахматы, или Почему вы не используете NetBIOS?

Команды NetBIOS

В данном разделе мы рассмотрим команды NetBIOS, употребление которых, по мнению автора, оправдано в прикладных программах. Три команды NetBIOS: RESET, GET ADAPTOR STATUS и SESSION STATUS здесь не рассматриваются, их подробное описание можно найти в [1]. Рассматриваемые в статье команды могут быть разбиты на три группы: команды поддержки имен, команды поддержки сеансов и команды поддержки дейтаграмм. Особняком стоит команда CANCEL, позволяющая прервать выполнение многих команд NetBIOS, выданных в режиме no-wait. Описание команд NetBIOS дается в следующем порядке: имя команды и ее код для режимов wait/no-wait, описание функций команды, описание входных данных с указанием полей NCB, используемых для их передачи NetBIOS, описание выходных данных с указанием полей NCB, в которых они передаются пользовательской программе, перечисление возможных кодов возврата. Для уменьшения повторений содержимое входных полей NCB_COMMAND и NCB_POST@ не оговаривается.

Команда CANCEL (35H)

Команда CANCEL предназначена для отмены команд NetBIOS, выданных в режиме no-wait. Сама команда CANCEL не может быть выдана в этом режиме. Ряд команд NetBIOS, перечисленных ниже, не может быть отменен командой CANCEL:

```
ADD NAME (30H/B0H)
DELETE NAME (31H/B1H)
ADD GROUP NAME (36H/B6H)
SEND DATAGRAM (20H/A0H)
SEND BROADCAST
   DATAGRAM (22H/A2H)
```

Успешная отмена команды SEND приводит к аварийному завершению сеанса.

Входные поля NCB:

NCB_BUFFER@ — поле должно содержать адрес NCB, связанного с отменяемой командой.

Выходные поля NCB:

Единственное поле — NCB_RETCODE, возможные коды возврата 0, 3, 22H, 23H, 24H, 26H, 4X, 50H-FEH.

Команды поддержки имен

ADD NAME (30H/B0H)

Команда добавляет уникальное имя в локальную таблицу имен.

Входные поля NCB:

NCB_NAME — должно содержать уникальное имя.

Выходные поля NCB:

NCB_NUM — в этом поле возвращается номер добавленного имени;

NCB_RETCODE — возможны следующие коды возврата: 0, 3, 0DH, 0EH, 15H, 16H, 19H, 21H, 22H, 23H, 4X, 50H-FEH.

ADD GROUP NAME (36H/B6H)

Команда заносит групповое имя в локальную таблицу имен.

Входные поля NCB:

NCB_NAME — в это поле помещается групповое имя.

Выходные поля NCB:

NCB_NUM — номер добавленного имени;

NCB_RETCODE — возможны следующие коды возврата: 0, 3, 0DH, 0EH, 16H, 21H, 22H, 23H, 25H, 4X, 50H-FEH.

DELETE NAME (31H/B1H)

Эта команда удаляет уникальное или групповое имя из локальной таблицы имен.

Входные поля NCB:

NCB_NAME — поле должно содержать удаляемое уникальное или групповое имя.

Выходные поля NCB:

Единственное поле — NCB_RETCODE — возможные коды возврата: 0, 3, 0FH, 15H, 21H, 22H, 23H, 4x, 51H-FEH.

Команды поддержки сеанса

CALL (10H/90H)

Команда CALL предназначена для установления сеанса с именем-партнером. Имя-партнер предварительно должно выдать команду LISTEN.

Входные поля NCB:

NCB_CALLNAME — поле должно содержать вызываемое имя-партнер;

NCB_NAME — поле должно содержать имя, открывающее сеанс;

NCB_RTO — в поле задается интервал таймаута для команд RECEIVE в единицах по 500 мс или 0, для отмены таймаута. Действует в течение сеанса;

Продолжение. Начало в КомпьютерПресс № 2'93.

NCB_STO — в данном поле задается величина интервала таймаута для команд SEND в единицах по 500 мс или 0 для отмены таймаута. Действует в течение всего сеанса.

Выходные поля NCB:

NCB_LSN — в этом поле возвращается номер сеанса, используемый командами SEND, RECEIVE и HANGUP;

NCB_RETCODE — возможны следующие коды возврата: 0, 3, 5, 9, 0BH, 11H, 12H, 14H, 15H, 18H, 19H, 1AH, 21H, 22H, 23H, 4X, 50H-FEH.

LISTEN (11H/91H)

Данная команда предназначена для установления сеанса с именем-партнером. Для нормального установления сеанса после выдачи этой команды имя-партнер должно выдать команду CALL.

Входные поля NCB:

NCB_CALLNAME — поле должно содержать имя-партнер по сеансу, если в первой позиции задана звездочка (*), установление сеанса возможно с любым именем, выдавшим команду CALL для данного имени;

NCB_NAME — это поле должно содержать имя, выдавшее команду;

NCB_RTO — поле задает величину интервала таймаута для команд RECEIVE, аналогично команде CALL;

NCB_STO — поле определяет величину интервала таймаута для команд SEND, аналогично команде CALL.

Выходные поля NCB:

NCB_CALLNAME — данное поле содержит имя, выдавшее команду CALL, что позволяет узнать имя-партнер в том случае, если оно было априорно неизвестно и было задано при выдаче команды LISTEN звездочкой (*);

NCB_LSN — в этом поле возвращается номер сеанса, используемый затем командами SEND, RECEIVE и HANGUP;

NCB_RETCODE — возможны следующие коды возврата: 0, 3, 9, 0BH, 11H, 15H, 17H, 18H, 19H, 1AH, 21H, 22H, 23H, 4X, 50H-FEH.

HANGUP (12H/92H)

Команда HANGUP предназначена для прекращения сеанса между двумя именами. Данная команда завершает все выданные в локальном узле, но еще незавершившиеся команды RECEIVE и RECEIVE ANY с кодом возврата 0AH (сеанс закрыт). Если есть незавершенная команда SEND, ей дается 20 с на завершение. Если за это время команда SEND завершается нормально, отменяется командой CANCEL, возникает таймаут или завершается сеанс в удаленном узле, команда HANGUP завершается нормально. Если команда HANGUP не завершается нормально в течение 20 с, она завершается с кодом возврата 5 (таймаут), при этом сеанс будет закрыт и все незавершенные команды RECEIVE (RECEIVE ANY) завершатся с кодом возврата 18H (аварийное завершение сеанса).

Входные поля NCB:

NCB_LSN — поле должно содержать номер сеанса, который идентифицирует закрываемый сеанс в локальном узле.

Выходные поля NCB:

NCB_RETCODE — возможны следующие коды возврата: 0, 3, 5, 8, 0AH, 0BH, 18H, 21H, 22H, 23H, 4X, 50H-FEH.

SEND (14H/94H)

Команда SEND передает заданное количество байтов данных из указанного буфера от одного имени другому имени, связанному с некоторым номером сеанса. Команда SEND должна завершиться до истечения интервала времени, который был задан в поле NCB_STO при открытии сеанса командами CALL/LISTEN. Если данные не будут переданы до истечения этого интервала, сеанс завершается аварийно, а команда SEND завершается с кодом возврата 5 (таймаут).

Входные поля NCB:

NCB_LSN — поле должно содержать номер сеанса;

NCB_BUFFER@ — поле должно содержать указатель на буфер;

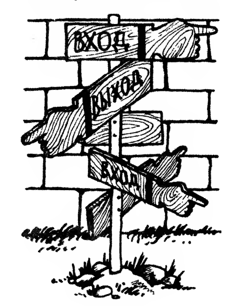
NCB_LENGTH — данное поле определяет длину передаваемых данных в байтах.

Выходные поля NCB:

NCB_RETCODE — возможные коды возврата: 0, 3, 5, 8, 0AH, 0BH, 18H, 21H, 22H, 23H, 4X, 50H-FEH.

CHAIN SEND (17H/97H)

Эта команда, так же, как и команда SEND, передает заданное количество



байтов данных из указанного буфера от одного имени другому, позволяя, однако, сцепить при передаче данные из первичного буфера с данными из вторичного буфера. В остальном команда CHAIN SEND ведет себя так же, как команда SEND.

Входные поля NCB:

NCB_LSN — должно содержать номер сеанса;

NCB_BUFFER@ — поле определяет адрес первичного буфера;

NCB_LENGTH — поле задает число байтов данных, передаваемых из первичного буфера;

NCB_CALLNAME — данное поле должно содержать длину данных (слово) и адрес вторичного буфера (два слова), остальные 10 байтов не используются.

Выходные поля NCB:

NCB_RETCODE — возможны следующие коды возврата: 0, 3, 5, 8, 0AH, 0BH, 18H, 21H, 22H, 23H, 4X, 50H-FEH.

RECEIVE (15H/95H)

Команда RECEIVE позволяет одному имени принять данные от другого имени, переданные посредством команд SEND и CHAIN SEND. Команда RECEIVE должна нормально завершиться до истечения интервала таймаута, который был задан в поле NCB_RTO при установлении сеанса командами CALL/LISTEN. Если данные не будут приняты до этого момента, сеанс не будет завершен аварийно, но команда RECEIVE завершится с кодом возврата 5 (таймаут).

Все незавершенные команды RECEIVE заканчиваются с ошибкой, если сеанс завершается аварийно или прекращается командой HANGUP.

Входные поля NCB:

NCB_LSN — поле должно содержать номер сеанса;

NCB_BUFFER@ — поле должно содержать указатель на приемный буфер;

NCB_LENGTH — данное поле определяет длину принимаемых данных (длину буфера);

Выходные поля NCB:

NCB_LENGTH — поле содержит действительную длину принятых данных;

NCB_RETCODE — возможны следующие коды возврата: 0, 3, 5, 6, 8, 0AH, 0BH, 18H, 21H, 22H, 23H, 4X, 50H-FEH.

RECEIVE ANY (16H/96H)

Команда RECEIVE ANY позволяет имени принять данные от любого из имен, с которыми у него был установлен сеанс, кроме того, она позволяет принять данные, адресованные любому имени в локальном узле сети. По завершении данная команда возвращает номер сеанса и номер имени. Это по-

Таблица 2. Коды возврата команд NetBIOS

Код возврата шест	Причина кода возврата и пояснения
00	Нормальное завершение команды
01	Неправильная длина буфера (попытка передать дейтаграмму с длиной более 512 байтов)
03	Неверный код команды
05	Таймаут (команда передачи/приема или установления сеанса не завершилась в течение заданного в полях NCB_STO или NCB_RTO интервала времени)
06	Сообщение не завершено (переданные данные не помещаются в буфере, длина которого была задана в поле NCB_LENGTH)
08	Неправильный номер сеанса (поле NCB_LSN содержит неправильный номер во время выдачи сеансовых команд SEND/RECEIVE)
09	Ресурс недоступен
0A	Сеанс закрыт (к моменту выдачи или до завершения сеансовой команды сеанс был закрыт)
0B	Команда отменена (команда, выполняющаяся в режиме no-wait, была отменена командой CANCEL)
0D	Дублирование имен (попытка добавить имя, уже содержащееся в локальной таблице имен)
0E	Локальная таблица имен заполнена (попытка добавить имя, когда в локальной таблице их уже 16)
0F	Имя имеет активный сеанс (попытка удалить имя, участвующее в сеансе)
11	Локальная таблица сеансов заполнена (попытка открыть слишком много сеансов)
12	Открытие сеанса отвергнуто (выдана команда CALL для имени, не выдавшего команду LISTEN)

зволяет ориентироваться среди многих сеансов и многих имен. Выполнение команды RECEIVE ANY зависит от содержимого поля NCB_NUM, как описано ниже. Если и команды RECEIVE и команды RECEIVE ANY поставлены в очередь на выполнение в рамках одного сеанса, они выполняются в следующем порядке: команды RECEIVE, затем RECEIVE ANY для конкретного сеанса и, в последнюю очередь, RECEIVE ANY для любого имени (если NCB_NUM=FFH).

Если сеанс завершается аварийно или прекращается по команде HANGUP, из незавершенных команд RECEIVE ANY для имен, связанных с данным сеансом, только одна команда RECEIVE ANY завершается с соответствующим кодом возврата.

Входные поля NCB:

NCB_NUM — это поле может содержать номер имени, выдавшего команду RECEIVE ANY или код FFH, для приема данных, переданных любому из имен в локальном узле сети;

NCB_BUFFER@ — поле должно содержать указатель на приемный буфер;

NCB_LENGTH — поле задает длину принимаемых данных в байтах.

Выходные поля NCB:

NCB_LSN — в этом поле возвращается номер сеанса, если данные приняты от имени, с которым был установлен сеанс;

NCB_NUM — если в этом поле был задан код FFH, то в нем возвращается номер имени, которому были адресованы данные;

NCB_LENGTH — в этом поле возвращается действительная длина принятых данных.

Команды поддержки дейтаграмм

Четыре команды, описываемые ниже, предназначены для обмена дейтаграммами. Напомним, что дейтаграммы представляют собой сообщения, прием которых в удаленном узле не гарантируется, но зато для их передачи не требуется открывать сеанс. Дейтаграммы могут быть посланы конкретному имени, групповому имени или направлены всем именам сети. Размер дейтаграмм ограничен 512 байтами.

SEND DATAGRAM (20H/A0H)

Эта команда предназначена для посылки дейтаграммы уникальному или групповому имени, как в локальном, так и в удаленном узле. Для приема дейтаграммы это имя должно заранее выдать команду RECEIVE DATAGRAM.

Входные поля NCB:

NCB_NUM — поле должно содержать номер имени, выдавшего команду;

NCB_BUFFER@ — данное поле должно содержать указатель на буфер с данными для передачи;

NCB_LENGTH — поле должно содержать длину передаваемых данных в байтах;

NCB_CALLNAME — данное поле должно содержать имя, которому направляется дейтаграмма.

Выходные поля NCB:

Единственное поле — NCB_RETCODE, возможные коды возврата: 0, 1, 3, 13H, 19H, 21H, 22H, 23H, 4X, 50H-FEH.

Некоторые эмуляторы NetBIOS при выполнении этой команды разрушают содержимое полей NCB_NAME и NCB_CALLNAME. Так, например, эмулятор NetBIOS LANsmart разрушает содержимое только одного поля NCB_CALLNAME, а эмулятор NetBIOS NetWare — обоих полей.

RECEIVE DATAGRAM (21H/A1H)

Данная команда предназначена для приема дейтаграмм, посланных посредством команды SEND DATAGRAM. Она не может использоваться для приема дейтаграмм, посланных командой SEND BROADCAST DATAGRAM. По завершении данная команда возвращает длину принятых данных и имя, которому дейтаграмма была послана. Выполнение команды зависит от содержимого входного поля NCB_NUM, как описано ниже.

Входные поля NCB:

NCB_NUM — это поле должно содержать номер имени, выдавшего данную команду или код FFH, для приема дейтаграммы, адресованной любому из имен в локальном узле сети;

Таблица 2. (Продолжение)

Код возврата шест	Причина кода возврата и пояснения
13	Недопустимый номер имени (поле NCB_NUM содержит неправильный номер)
14	Имя не найдено или нет ответа (имя, вызываемое по команде CALL, либо отсутствует в сети, либо не отвечает)
15	Имя не может содержать в первой позиции 0 или *
17	Имя удалено (попытка использовать имя удаленное из таблицы имен)
18	Аварийное завершение сеанса
19	Конфликт имен
1A	Несовместимое удаленное устройство
21	Интерфейс занят
22	Слишком много команд поставлено в очередь (количество команд, запущенных в режиме no-wait, превысило лимит, заданный при запуске эмулятора NetBIOS или по умолчанию)
23	Неправильный номер сетевого адаптера в поле NCB_LANA_NUM
24	Команда уже нормально завершилась к моменту выдачи команды CANCEL
26	Команда не может быть отменена командой CANCEL
4x	Необычные условия в сети, x -- любая шестнадцатиричная цифра
50 - FE	Сбой сетевого адаптера

Примечание: пояснения в скобках даны автором статьи.

NCB_BUFFER@ — в этом поле должен находиться указатель на буфер для приема данных;

NCB_LENGTH — поле должно содержать длину приемного буфера в байтах.

Выходные поля NCB:

NCB_LENGTH — поле содержит длину принятой дейтаграммы;

NCB_CALLNAME — в этом поле возвращается имя, которому была передана данная дейтаграмма;

NCB_RETCODE — возможные коды возврата: 0, 3, 6, 0BH, 13H, 17H, 19H, 21H, 22H, 23H, 4X, 50-FEH.

SEND BROADCAST DATAGRAM (22H/A2H)

Эта команда используется для посылки дейтаграмм всем существующим в сети именам, которые выдали команду RECEIVE BROADCAST DATAGRAM. Если в локальном узле есть незавершенные команды RECEIVE BROADCAST DATAGRAM, они завершатся, приняв дейтаграмму. Заметим, что несколько команд RECEIVE BROADCAST DATAGRAM, требующих приема дейтаграммы (выданных разными именами), удовлетворяются единственной командой SEND BROADCAST DATAGRAM.

Входные поля NCB:

NCB_NUM — поле должно содержать номер имени, выдавшего команду;

NCB_BUFFER@ — данное поле должно содержать адрес буфера с данными для передачи;

NCB_LENGTH — это поле должно содержать длину передаваемых данных в байтах.

Выходные поля NCB:

NCB_RETCODE — возможные коды возврата: 0, 1, 3, 13H, 19H, 21H, 22H, 4X, 50H-FEH.

RECEIVE BROADCAST DATAGRAM (23H/A3H)

Эта команда используется для приема дейтаграмм, разосланных посредством команды SEND BROADCAST DATAGRAM. Команда не может использоваться для приема дейтаграмм, посланных конкретному имени по команде SEND DATAGRAM. По завершении команда возвращает длину принятой дейтаграммы и имя, ее пославшее.

Входные поля NCB:

NCB_NUM — поле должно содержать номер имени, выдавшего команду;

NCB_BUFFER@ — данное поле должно содержать указатель на приемный буфер;

NCB_LENGTH — это поле должно содержать размер приемного буфера в байтах.

Выходные поля NCB:

NCB_LENGTH — в этом поле возвращается длина принятых данных;

NCB_CALLNAME — в данном поле возвращается имя, передавшее дейтаграмму;

NCB_RETCODE — возможны следующие коды возврата: 0, 3, 6, 0BH, 13H, 17H, 19H, 21H, 22H, 23H, 4X, 50H-FEH.

В.Баевский

(Продолжение следует)

Использованные источники:

1. LANsmart Programmer's Reference, by D-Link Inc. — Datex Inc., 1989.
2. Brown Ralf, Interrupt list, Release 91.1, Last change 1/5/91.
3. IntrList version 1.02, by Window Book, Inc., 1990.
4. LAN Evaluation Report, 1986, by Novell Inc., Orem, Utah.
5. Mier Edwin, Controlling protocol confusing, перевод статьи в Computer World Moscow под названием "Как разобраться с протоколами", № 13, 1992.
6. Tech Help version 4.02, by Flambeux Software, 1990.



Sierra: более 10 лет в индустрии компьютерных игр

Не будет преувеличением сказать, что игры фирмы Sierra-OnLine занимают первое место по популярности в нашей стране. Знаменитые серии King Quest, Space Quest, Police Quest и Larry отнимали и отнимают у нас значительную часть рабочего и нерабочего времени. Игры фирмы Sierra прошли значительный путь от простых игр, умещавшихся на одну дискету объемом 360 Кбайт, до современных, сочетающих в себе анимацию, синтезированную речь и музыкальное сопровождение и занимающих десятки мегабайт. Основной задачей фирмы было противопоставление компьютерных игр телевидению. Несмотря на и по сей день существующие аппаратные ограничения, игры фирмы Sierra все блнже и ближе подходят к выполнению этой задачи. Сочетанне графики высокого разрешения, выполненной профессиональными художниками, анимации, синтезированной речи и музыкального сопровождения, дополненное увлекательным сюжетом, заставляет нас часами, днями, месяцами проводить время вместе с королем Грэхемом, Роджером Вилко, Джимом Уолсом и другими не менее известными персонажами.

Сделав ставку на компьютерные игры, Sierra не только предлагает нам бездумные развлечения, но и способствует процессу обучения. Такие продукты, как Mixed Up Mother Goose, Dr. Brain и Island of Dr. Brain, предназначены для школьников и способствуют постижению разлнчных областей знаний. Фирма реализует комплекты этих игр с пособиями для учителей, что помогает их использованию в школьном процессе. В ближайшее время эта серня пополнится играми, посвященными основам физики и истории США.

Диапазон игр, выпускаемых фнрмой сегодня, достаточно широк: это и игры жанра “фэнтази“, детективы, триллеры, игры с “клубничкой“, и динамические игры типа “бей все, что шевелится“. Объединение усилий со многими фирмами из Европы, Японии и США дает фирме больше возможностей для расширения сферы выпускаемой игровой продукции.

Одна из последних разработок фирмы ImagiNation — многопользовательская сетевая игра, доступная через модем.

Подводя итоги 1992 года, президент фирмы Кен Вильямс отмечал, что компьютерные игры все ближе и ближе подходят к видео- и кинопродукции. Применение последних технических достижений: компактных дисков (CD), MTV и графики высокого разрешения, а также сценариев, не уступающих художественным произведениям, позволяет говорить еще об одном виде искусства — искусстве компьютерных игр.

10 самых популярных игр (январь 1993)

1. King's Quest VI: Heir Today, Gone Tomorrow
2. Quest for Glory III: Wages of War
3. Aces of the Pacific
4. Laura Bow in the Dagger of Amon Ra
5. Red Baron Mission Builder
6. King's Quest V: Absence Makes the Heart Go Yonder
7. Quest for Glory I: So You Want To Be A Hero
8. Police Quest III: The Kindred
9. Gobliins
10. Take-A-Break! Crosswords

ЧТО НАС ОЖИДАЕТ?

Как всегда, любители игр с интересом ждут новостей от фирмы. Фирма Sierra On-Line объявила о планируемом в первом полугодии 1993 года выпуске следующих игр:

Aces Over Europe

Игра, возвращающая нас в период Второй мировой войны. Европейский театр военных действий. Танки, поезда, самолеты — одним словом, в войне побеждает сильнейший. Планируется к выпуску весной.

Codename: Phoenix

Шпионский триллер, местом действия которого станут многие страны. Планируется к выпуску в конце весны.

EcoQuest: Lost Secrets of the Rainforest

Продолжение известной серии EcoQuest: The Search for Cetus. Множество новых приключений, речевые вставки и кнопка Help, помогающая в безвыходных ситуациях. Планируется к выпуску в начале 1993 года.

Freddy Pharkas: Frontier Pharmacist

Новый проект создателя известной серии игр Larry Эла Лоу. Комедия из жизни Дикого Запада. Планируется к выпуску весной.

Front Page Sports: Baseball

По сообщениям фирмы, новый подход к реализации позволит играющим "наблюдать ход бейсбольной игры" по телевизору. Планируется к выпуску в начале года.

Gobliins 2

Более "крутая" версия широко известной и популярной игры про трех друзей-гоблинов. В данной версии выступает дуэт: Фингус-дипломат и Винкл-шутник. Планируется к выпуску в начале 1993 года.

Inca

Приключенческая игра из жизни инков с фантастическим сюжетом. Прекрасная графика и многочисленные загадки доставят истинное удовольствие любителям приключенческих игр. Планируется к выпуску в начале 1993 года.

Incredible Machine

Симпатичная игра, в которой необходимо либо создавать собственные машины из набора деталей, либо достраивать уже готовые.

Police Quest IV

Новая версия этой игры находится в стадии разработки и ее появление возможно к концу 1993 года.

RAF In The Pacific

Вторая мировая война, воздушные бои. Возможно использование самолетов из популярной игры Aces of the Pacific.

Space Quest V

Продолжение космических приключений Роджера Вилко. Как обычно, множество приключений, иные миры, прекрасная графика. Планируется к выпуску в начале 1993 года.

Turbo Science

Игра с элементами обучения, весело и доходчиво раскрывающая основные законы физики.

Raymond E.Feist's Riftwar Legacy, Vol. One: Betrayal At Krondor

Приключения и фантастика. Фантастический мир занимает более 224000 футов. Планируется к выпуску в начале 1993 года.

Slater & Charlie Go Camping

Игра для детей, знакомящая с приключениями двух маленьких динозавров. Все слова, появляющиеся на экране, озвучиваются синтезатором речи, что облегчает детям игру и учит их произношению. Планируется к выпуску летом 1993 года.

Twisty History

Перепутанная история: занимательно и полезно. Путешествие во времени в компании известных исторических личностей. Планируется к выпуску в начале 1993 года.

World War II: 1946

Дополнительный игровой диск к Aces of the Pacific. Война в воздухе над японскими островами.

Take-A-Break! Pinball

Игра для среды Microsoft Windows, планируется к выпуску во второй половине 1993 года.

Проекты...

Новую версию игры Larry 6 можно ожидать не ранее Рождества 1993 года. Пока не планируется продолжение таких бестселлеров, как Heart of China и Rise of Dragon, но рассматривается возможность создания игры, совмещающей в себе элементы мистики, ужасов и фантастики. Также пока не планируется выпуск новой версии King Quest VII.

А.Федоров

По материалам, предоставленным фирмой Sierra On-Line

"Любое программное обеспечение хорошо ровно настолько, насколько хороша фирма, стоящая за ним. Этот урок рано или поздно выучивает каждый покупатель. И часто учение дается слишком болезненно.
Так бывает, когда поставщик программной продукции вдруг перестает заботиться о сервисе и поддержке, перестает вкладывать деньги в исследования и разработку, перестает заботиться о качестве и постепенно покидает бизнес.
В жестоком, полном конкуренции мире высоких технологий такое случается постоянно.
Вот почему покупатель должен всегда смотреть шире, чем просто на достоинства того или иного продукта. При покупке нужно оценивать не только программное обеспечение, но и компанию, его производящую.
Любую компанию можно быстро оценить, задав прямой вопрос о ее финансовой стабильности, сервисе, поддержке, вложениях в исследования и разработку.
Фирма Computer Associates приветствует подобные вопросы. Сейчас вы поймете, почему."

Чарльз Б. Вонг,
президент и главный исполнительный директор



Фирма и люди

Какие ассоциации вызывают слова *Computer Associates*? У тех, кто знаком с фирмой СА, в первую очередь — программное обеспечение.

В СА мы не продаем "железо", периферию или сети. Мы делаем программы — ничего больше.

От системного программного обеспечения до программ работы с информацией и деловых приложений — мы разрабатываем, производим, поддерживаем, улучшаем и расширяем поистине все виды существующей программной продукции. Для компьютеров всех типов: от больших до мини- и микроЭВМ. Для компьютеров всех распространенных в мире марок: IBM, ICL, Digital, Unisys, Apple, Hewlett-Packard, Bull, Tandem, Data General, Hitachi, Fujitsu и многих других.

Сегодня мы предлагаем более 300 программных решений, за которыми — тысячи талантливых, опытных, увлеченных профессионалов. Все они работают над тем, чтобы сделать программы СА эффективнее, а покупателю принести успех.

Computer Associates создает программы для применения как в маленьком домашнем офисе, так и в больших фирмах. Успех сотен тысяч компаний во всем мире зависит от программ СА. Более 90% из 500 крупнейших компаний мира пользуются нашими программами. И это при том, что фирма Computer Associates ведет свою историю лишь с 1976 года. Именно тогда Чарльз Б. Вонг, нынешний президент и главный исполнительный директор фирмы, с тремя компаньонами основал СА.

С тех пор мы очень выросли. Более 8 тысяч сотрудников Computer Associates работают в более чем 100 отделениях фирмы в 26 странах мира. И с самого начала мы были настроены на то, чтобы делать лучшие программы в мире.

Наши клиенты получают всестороннюю поддержку и помощь

Наш подход к обслуживанию и поддержке клиентов очень прост. Ваши проблемы — это наши проблемы. Быстро и эффективно решать их помогает разветвленная сеть поддержки, действующая по всему миру. В наших отделениях на местах работают тысячи квалифицированных специалистов. Каждое отделение имеет экспертов по системному программному обеспечению, программам работы с информацией и деловым приложениям.

Мы первыми внедрили прогрессивные системы поддержки On-Line (такие как CA-TCC, Total Client Care), позволяющие связываться с информационными базами Computer Associates, реализованными на больших ЭВМ. Это дает возможность пользователю находить быстрый ответ на любой вопрос по работе с программами СА и быстро решать любые проблемы. Тем же способом вы можете связаться с персоналом СА и получить любую интересующую вас информацию. При помощи системы On-Line пользователю доступны новейшие методы обучения, базирующиеся на технологии мультимедиа.

Мы проводим обширные программы обучения пользователей. Нами созданы тысячи специализированных классов, семинаров, организуются регулярные встречи групп пользователей, налажен выпуск специальных обучающих видеокассет. А служба поддержки и сопровождения предлагает разнообразные сервисные услуги, в том числе установку, наладку и запуск в эксплуатацию продукции CA.

Мы вкладываем больше средств в исследования и разработку, чем другие фирмы

3,4 млн. долл. в неделю. Каждую неделю. Более 170 млн. долл. в год тратится только на исследования и разработку программного обеспечения.

Только за прошлый год мы создали сотни усовершенствованиых версий наших программных продуктов. Мы разработали также десятки совершенно новых продуктов, в которых нашли воплощение новейшие технологии, помогающие решать сложнейшие проблемы сегодняшнего дня.

Решение проблем — вот наша задача. И именно эту задачу мы ставим перед тысячами программистов, инженеров и разработчиков, работающих в Центрах разработки программного обеспечения CA по всему миру. Мы требуем, чтобы они внимательно выслушивали пользователей и выясняли их проблемы, а затем создавали для них соответствующие программные решения. Мы направляем специальные группы разработчиков на исследования общих проблем и тенденций в промышленности, таких как интеграция, безопасность, минимизация и т.д.

Эффективность наших усилий в исследованиях и разработке значительно возрастает благодаря давнему и тесному сотрудничеству с IBM, а также уникальным стратегическим альянсам с Digital, Hewlett-Packard, Apple и Tandem.

CA90s — архитектура 90-х

Сегодня вам нужно нечто большее, чем просто подходящее "железо", и большее, чем просто подходящие программы. Вам нужны правильные комбинации — платформ, операционных систем и приложеиий.

Но как сделать правильный выбор во всем многообразии существующих аппаратных решений и операционных систем? Ответ в выборе правильной архитектуры программного обеспечения — наша архитектура, архитектура CA90s.

CA90s не имеет каких-либо аппаратных пристрастий, она не привязана ни к какой специфической платформе или операционной системе. CA90s предоставляет обширнейший спектр программных решений для широчайшего диапазона аппаратных платформ и операционных систем. За вами остается свобода выбора, смены, наращивания. CA90s предлагает поддержку SAA, NAS и других распространенных промышленных стандартов — CA90s объединяет и расширяет их.

Поэтому, когда вы ищете решения по автоматизации, интеграции, минимизации — обратите внимание на преимущества, предлагаемые архитектурой CA90s. Они поистине всеобъемлющи. И они всегда доступны.

Качество — наша главная забота

Независимая лаборатория качества Computer Associates — это лучшие специалисты компьютерной индустрии. И они неустанно работают над улучшением качества Computer Associates — качества нашей продукции, сервиса, поддержки, качества всего, что мы делаем.

Профессионалы в области программного обеспечения и эксперты по организации и управлению производством ведут исследования по работе Computer Associates. Они выясняют сильные и слабые, хорошие и плохие стороны, определяют цели, ставят задачи, утверждают высокие стандарты. Затем дают свои рекомендации, обязательные для выполнения.

Нужно ли говорить, что не всегда их предложения нам по душе. Но так или иначе — все это помогает нам становиться лучше и сильнее. Мы очень стараемся. А они говорят нам, как это делать.

Сегодня вам нужно нечто большее, чем подходящее программное обеспечение. Вам нужен подходящий деловой партнер

В мире существует более 30 000 компаний, заиимающихся программным обеспечением. Но среди них вы не найдете лучшего партнера, чем Computer Associates. За нами — финансовая стабильность и гарантии безопасных капиталовложений фирмы, стóящей полтора миллиарда долларов.

Мы вкладываем более 170 млн. долл. в год в исследоваиия и разработку. Мы делаем больше разных видов программного обеспечения для большего количества типов компьютеров, чем кто-либо другой в мире. Мы имеем самую разветвленную систему сервиса и поддержки в мировой индустрии программного обеспечения. И мы создали самую развитую и всеобъемлющую программную архитектуру, когда-либо существовавшую в мире.

Но самая главная причина правильиости выбора CA — не техническая, а человеческая. Наши люди — это наше будущее. Их талант, ум и опыт — вот что выводит вперед нашу технологию. Их увлеченность и профессионализм, обязательность и верность нашим клиентам делают Computer Associates не просто компанией по производству программного обеспечения, а вашим надежным деловым партнером. Партнером, на которого можно рассчитывать. Всегда. Везде.

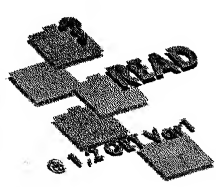
Позвоните нам — и мы к вам придем.





Фейерверк программных продуктов

CA-dBFast for Windows



CA-dBFast — это первая и единственная полная, автономная, dBASE-совместимая система для Microsoft Windows. Она позволяет быстро и легко создавать мощные графические базы данных, работающие в среде Windows.

- **Все преимущества Windows для пользователей dBASE.** Графические, многоцветные приложения CA-dBFast, использование окон, меню, диалоговых окон, кнопок, растровых изображений и т.д.
- **Ваши усилия, затраченные на программирование, не пропадут даром.** Любые dBASE III Plus-совместимые программы будут без переделок работать под Windows. Несколько простых манипуляций — и вы безболезненно перенесете свои программы в CA-dBFast.
- **Мощная языковая поддержка.** Разработчики на dBASE III Plus, dBASE IV, CA-Clipper и FoxBase найдут здесь все возможности, необходимые для создания развитых Windows-приложений.
- **Связь с другими Windows-приложениями.** Полная поддержка динамически связываемых библиотек (DLL) и динамического обмена данными (DDE) Windows. Обмен данными между приложениями CA-dBFast и другими Windows-приложениями. Поддержка любых NetBIOS-совместимых локальных сетей, включая Novell, LAN Manager, Token Ring и др.

CA-Clipper for DOS

CA-Clipper — это развитый язык программирования, гибкий препроцессор, высокопроизводительный компилятор и эффективный компоновщик. Совместно с редактором, отладчиком и утилитой make CA-Clipper представляет собой законченную систему создания профессиональных PC-приложений, в том числе ориентированных на работу в сетях. CA-Clipper предлагает открытую среду программирования с мощной поддержкой сетей и баз данных.

- **Создание автономных приложений.** CA-Clipper создает самостоятельные прикладные программы в виде исполняемых EXE-файлов, для работы которых не требуется дополнительного программного обеспечения.
- **Открытая архитектура, свобода для наращивания.** Определяемые пользователем команды и функции; возможность создания собственных библиотек; импорт модулей, написанных на языках C, Assembler, dBASE, Pascal; подключаемые драйверы баз данных.
- **Выход за пределы барьера памяти DOS.** Динамическое управление оверлеями позволяет создавать приложения, превышающие размером доступную оперативную память, при этом нет необходимости в ручном создании статических оверлеев. Менеджер виртуальной памяти позволяет приложениям использовать до нескольких мегабайт строк и массивов.
- **Драйвер баз данных формата DBF.** 1 млрд. записей в файле данных; 1000 полей в записи; более 200 одновременно используемых файлов данных; 15 активных индексов на файл данных; числовые поля из 30 цифр с точностью до 16 значащих цифр; поддержка логических полей; memo-поля длиной до 65 536 символов.
- **Существует русская версия.**

CA-Clipper Tools II for DOS

CA-Clipper Tools II — это обширнейшая библиотека инструментальных средств для CA-Clipper. CA-Clipper Tools II предлагает готовые эффективные решения общих проблем в прикладных задачах. Сконфигурированная и оптимизированная для поддержки профессиональных разработчиков, библиотека CA-Clipper Tools II открывает перед программистами CA-Clipper новые области прикладных решений, которые раньше требовали применения С или ассемблера.

- **Набор проверенных решений.**
- **17 категорий функций.** Вам нужно закрасить экран, изменить размер окна или проверить активность сети. CA-Clipper Tools II дает разработчику CA-Clipper мощную поддержку в лице более чем 550 функций, разбитых на 17 категорий. От превосходных видео- и экранных функций до дисковых и сетевых утилит CA-Clipper предлагает проверенные, эффективные решения общих прикладных проблем.
- **Разнообразие систем помощи.** Множество примеров программ и специальные утилиты помогут быстро продвинуться в освоении пакета, предлагая простые и наглядные образцы использования функций CA-Clipper Tools II или давая необходимую справочную информацию.
- **Существует русская версия.**

CA-Clipper/Compiler Kit

CA-Clipper/Compiler Kit — это развитый компилятор dBASE IV-приложений. Компилируя приложения dBASE IV с помощью CA-Clipper/Compiler Kit, вы получаете массу преимуществ, доступных ранее только разработчикам на CA-Clipper. CA-Clipper/Compiler Kit создает автономные исполняемые файлы из готовых dBASE IV-программ.

- **Нет нужды в опыте программирования на CA-Clipper.** Поддержка в CA-Clipper определяемых пользователем команд используется для имитации поведения dBASE IV. Затем препроцессор и компилятор CA-Clipper компилируют перетранслированный код. Можно компилировать программы, полученные при помощи генератора приложений dBASE IV. Скомпилированные приложения сохраняют полную функциональность dBASE IV, приобретая превосходные свойства управления памятью CA-Clipper и великолепную производительность. Приложения полностью совместимы с dBASE IV-файлами .DBF, .MDX и .DBT. Полностью поддерживаются отчеты, этикетки и запросы dBASE IV. По желанию можно использовать все возможности настраиваемого языка CA-Clipper и классы объектов.
- **Четыре компонента автоматизируют процесс компиляции.** CA-Clipper/Compiler Kit for dBASE IV включает: препроцессор, настраивающий header-файл, библиотеку определяемых пользователем функций, заменяемый драйвер баз данных .MDX. Препроцессор копирует .PRD-файл и, используя инструкции настраивающего header-файла, построчно проходит по файлу dBASE IV, преобразуя команды интерпретатора в код, пригодный для компиляции CA-Clipper. Библиотека определяемых пользователем функций используется для дополнения стандартных функций CA-Clipper и имитации поведения dBASE IV, а заменяемый драйвер баз данных призван обеспечить управление dBASE IV-совместимыми файлами.
- **Автоматическое управление распределением памяти без всякого программирования.** Динамический менеджер оверлеев позволяет использовать приложения, превышающие размером доступную оперативную память, при этом нет необходимости в ручном создании статических оверлеев. Менеджер виртуальной памяти CA-Clipper позволяет приложениям использовать до нескольких мегабайт строк и массивов.
- **Высочайшая производительность и защита исходных текстов программ.** Программы работают на всех сетях, поддерживающих DOS 3.1 и выше. Высокая производительность и надежность, в том числе при работе в сетях. Скомпилированные приложения обеспечивают защиту исходных текстов программ.
- **Существует русская версия.**

CA-REALIZER

Полная система разработки BASIC-приложений для Windows. CA-REALIZER объединяет структурный набор команд языка BASIC, расширенный для доступа к объектам и ресурсам Windows, среду визуально-интуитивной разработки программ и набор профессиональных средств создания Windows-приложений Programmable Application Tools.

- **Интерактивная среда разработчика позволяет наглядно управлять процессом создания программ.** Пошаговое выполнение кода программы, трассировка, просмотр дерева вызова процедур, просмотр любых локальных или глобальных переменных, прогонка любой части программы из отладчика. Моментальная смена

интерфейсных объектов, шрифтов, цветов с просмотром результатов изменений. Генерация с помощью Project Builder исполняемого EXE-модуля, использующего в своей работе runtime-библиотеку CA-REALIZER.

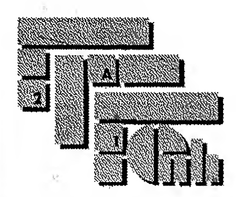
- **Профессиональный набор средств создания Windows-приложений Programmable Application Tools.** Простые команды позволяют встраивать в разрабатываемые программы элементы профессиональных Windows-приложений: диаграммы, электронные таблицы, текстовые редакторы, анимацию, графические таблицы и пользовательские формы.
- **Визуально-интуитивное создание интерфейса.** Возможность "рисовать" на экране объекты Windows — кнопки, текстовые поля, списки, растровые изображения — и встраивать их в разрабатываемую программу без единой строчки кода.
- **Использование модулей, написанных на C, Pascal и других языках.** Расширение окружения CA-REALIZER при помощи динамически связываемых библиотек (DLL). Подключение баз данных, эмуляторов терминалов, видео- и мультимедиа-дисплеев, средств сбора данных и деловой графики.
- **Использование развитых возможностей Windows.** Поддержка динамического обмена данными (DDE), работа с последовательным портом, доступ к любым функциям Windows API, поддержка нестандартных элементов управления, связь с другими инструментальными средствами CA-REALIZER, запуск DOS- и Windows-команд в указанное время и с заданными интервалами. Поддержка большого числа различных файловых форматов: Lotus 1-2-3, Excel, BMP, CA-Textor, CA-SuperCalc и др.
- **Высокая совместимость с DOS-версиями BASIC.** Совместимость с широко распространенными версиями языка BASIC, в том числе с Microsoft QuickBASIC, позволяет легко переносить готовые DOS-программы в Windows.

CA-Textor for Windows

Текстовый процессор, разработанный специально для тех, кому нужны высококачественные печатные материалы, но кто не хочет тратить время на освоение громоздких и перегруженных издательских систем.

- **Простота в использовании.** CA-Textor предлагает простой, дружественный пользовательский интерфейс со всеми возможностями Windows-ориентированных систем и развитой контекстно-чувствительной помощью.
- **Мощные возможности работы с текстами.** Подключаемые пользовательские словари с проверкой орфографии и словарями синонимов, редактирование в текстовом и в WYSIWYG-режимах, использование MDI-интерфейса для одновременной работы с несколькими документами, встроенная графика, развитый табличный редактор, программируемый калькулятор и многое другое.
- **Полный контроль над редактируемым документом.** Пользовательский глоссарий, вставка рисунков и таблиц (файлы форматов CGM, TIFF, WFF, WPG и BMP), многооконное редактирование, форматирование и автоматическая нумерация страниц, таблица стилей, колонтитулы, просмотр в графическом режиме с увеличением, поиск и замена, структурное редактирование текста, автоматическая генерация содержания, поддержка динамического обмена данными, автоматический перенос слов, импорт и экспорт файлов формата WordPerfect, Word, DCA/DIA (RTF) и ASCII, поддержка разнообразных принтеров (включая PostScript).
- **Существует русская версия.**

CA-SuperCalc for DOS



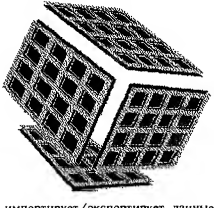
Популярнейшие электронные таблицы CA-SuperCalc сочетают простоту в использовании с уникальной мощью развитого трехмерного моделирования, связывания электронных таблиц, построения разнообразных диаграмм, печати электронных таблиц, реляционной базы данных и всестороннего аудирования.

CA-SuperCalc — это серьезные электронные таблицы, разработанные с тем расчетом, чтобы даже новичок мог в полной мере использовать все возможности своего компьютера.

- **Простота в использовании.** Полная поддержка мыши, ToolBar — панель быстрого доступа к типовым операциям, быстрое построение диаграмм, Function Assistant, Fast Preview и т.д.
- **Наглядное отображение информации на экране и принтере.** Построение самых разнообразных цветных диаграмм, в том числе с использованием трехмерной графики, в режиме WYSIWYG.

- **Связывание электронных таблиц и отдельных страниц.** CA-SuperCalc позволяет связывать до 255 электронных таблиц и отдельных страниц в памяти или на диске.
- **Отчеты и графика презентационного качества.** Независимо от типа принтера вы можете использовать разнообразные шрифты, рамки, отбивки, заливки и т.д. Предварительный просмотр в графическом режиме с масштабированием. Переключение портретной/ландшафтной ориентации изображения для бумаги любых размеров. Автоматическое масштабирование изображения под размер бумаги.
- **Встроенные средства отладки.**
- **Совместимость с файлами и макросами Lotus 1-2-3.**
- **Работа в локальных сетях.**
- **Встроенная реляционная база данных Silverado.**
- **Существует русская версия.**

CA-Compete! for Windows



CA-Compete! объединяет электронную таблицу, средства для работы с базами данных и для многомерного моделирования.

- **Простота в использовании.** CA-Compete! использует все преимущества графического интерфейса Windows. Схожесть с традиционными электронными таблицами облегчает освоение пакета.
- **Объектная ориентация.** Объектно-ориентированные ссылки на ячейки позволяют использовать любую удобную пользователю терминологию.
- **Многомерность.** До 12 измерений или категорий просматриваемых данных в представлении электронных таблиц. Мгновенные повороты многомерной модели под любой угол зрения мышью.
- **Глобальные формулы.** Распространение формул на выбранные ячейки по всем измерениям созданной модели.
- **Мощные средства обработки и отладки.**
- **Высококачественная печать.**
- **Использование данных различных форматов.** CA-Compete! импортирует/экспортирует данные форматов CA-SuperCalc, Lotus 1-2-3 (WKS/1), Microsoft Excel (XLS), ASCII (TXT, CSV), dBASE (DBF) и script-файлы. Поддерживается динамический обмен данными (DDE) с другими Windows-программами.

(Окончание следует)



Знакомьтесь: фирма TurboPower Software

Фирма TurboPower Software основана в 1985 году Кимом Кокконеном (Kim Kokkonen), который до этого более 11 лет сотрудничал с различными фирмами, включая и фирму Intel. Кокконеном был создан и первый продукт фирмы — TurboPower Utilities — набор утилит для пользователей компилятора Turbo Pascal. Первым же изделием, завоевавшим популярность у программистов, был продукт под названием Turbo Extender, позволявший создавать программы большой модели памяти (large memory model) с помощью компилятора Turbo Pascal 3. Также был выпущен программный продукт T-DebugPlus — первый символьный отладчик для компилятора Turbo Pascal и Turbo Optimizer — оптимизатор кода.

Для компилятора Turbo Pascal версии 4.0 были выпущены библиотека Turbo Professional (явившаяся прообразом библиотеки Object Professional), новая версия отладчика — T-DebugPlus, а также средство управления оверлеями. Позже эти продукты были реализованы для версии Turbo Pascal 5, тогда же был выпущен набор подпрограмм для создания баз данных (B-Tree Filer). С появлением версии компилятора Turbo Pascal 5.5 стала возможной разработка объектно-ориентированной библиотеки Object Professional. Интересно отметить, что библиотека Object Professional была выпущена на 1,5 года раньше библиотеки Turbo Vision.

Для компилятора Turbo Pascal были также разработаны Async Professional — набор подпрограмм и объектов для поддержки коммуникаций, TSR Made Easy — библиотека для создания резидентных программ и Turbo Analyst — набор утилит для программистов, являющийся расширением TurboPower Utilities.

Недавно фирма TurboPower выпустила библиотеку Win/Sys, предназначенную для программистов, работающих в среде Microsoft Windows, а также средство для создания пользовательских интерфейсов и элементов управления Data Entry Workshop.

Оставаясь в первую очередь разработчиком библиотек для компилятора Turbo Pascal, фирма TurboPower начала перенос своих продуктов в среду C/C++. Наряду с уже выпущенными версиями Object Professional для C++ и TSRs and More в этом году планируется выпуск версий библиотек B-Tree Filer и Async Professional для компиляторов C/C++. Продукты для среды Windows построены на основе динамически загружаемых библиотек, они доступны как для компилятора Pascal, так и для С-компилятора.

В настоящее время в TurboPower работает 8 человек. Высокой продуктивностью фирма обязана сплоченности коллектива и тесным дружеским контактам с другими компаниями. Так, фирма Enz Edv Beratung GmbH из Германии ведет исследования и разработки продукта B-Tree Filer, фирма Techmate, Inc. (Флорида, США) выполнявшая перенос большинства продуктов фирмы TurboPower в среду C/C++, занимается их техническим сопровождением.

Одной из отличительных черт фирмы TurboPower является прекрасно организованная техническая поддержка, осуществляемая через CompuServe, FidoNet и собственную BBS. Фирма TurboPower находится в тесном контакте с фирмой Borland; это позволяет им выпускать новые версии своих библиотек практически одновременно с выходом новых версий компилятора.

А.Федоров



Мультимедиа в трех измерениях

Amiga и VideoToaster

Когда после PC Magazine, PC World и им подобных (где до сих пор материалы, относящиеся к мультимедиа, занимают достаточно небольшой процент общего объема) берешься читать MacWorld или MacUser, то сразу начинаешь тонуть в обилии "мультимедийной" информации и рекламы: анимация, 3D-графика, работа с 24-битными изображениями высокого разрешения, всевозможные авторские системы, системы гипермедиа, видео- и аудиоплаты; и QuickTime, QuickTime везде и во всем... Такая интегрированная "весовая" оценка "платформо-ориентированных" компьютерных журналов достаточно адекватно отражает степень проникновения технологий мультимедиа на компьютеры разных линий. Так вот, при чтении многочисленных журналов "мира Amiga" — Amiga World, Amiga World Tech, Amiga Computing International, Amiga User, CU Amiga и др. — бросается в глаза все усиливающаяся ориентация этого мира именно на видео, в том числе и видео профессионального уровня. Страницы этих журналов полны специфических терминов и понятий из области видео и ТВ, незнакомых обычным пользователям персональных компьютеров; доля посвященных видеопроизводству материалов все время увеличивается, растет число описываемых и рекламируемых программных и аппаратных продуктов для этой области деятельности, все чаще появляются руководства по построению полной конфигурации видеостудии.

Продолжение. Начало в КомпьютерПресс №№ 1,2'93.

Благодаря техническим особенностям компьютеров Amiga, их "привязке" к видеостандартам (см. КомпьютерПресс № 7'91), стыковка с видеоаппаратурой требует значительно меньших усилий на аппаратном и программном уровнях; сравнительно невысока и стоимость самих компьютеров. Поэтому студия на базе Amiga может оказаться вдвое—втрое дешевле студии на базе Mac или IBM при почти сопоставимых возможностях. В США, по данным 1991 года, 2/3 фирм, предоставляющих услуги в области видеопроизводства и "пост-производства", использовали в своей работе системы на базе Amiga. Выпуск Amiga-3000 на 68030 ценой около 3000 фунтов стерлингов (прошу извинить за смесь валют — я привожу без изменений цены из разных источников), с памятью, расширяемой до 32 Мбайт, плат-акселераторов для моделей 1500 и 2000 на 68030/68882/50 МГц (1800 фунтов стерлингов), 68040/50 (1700 фунтов стерлингов с 8-Мбайт RAM), совсем недавнее появление Amiga 4000 — все это должно еще более упрочить позиции Amiga на этом рынке, обеспечить, помимо имеющихся достоинств, еще и возможность достаточно быстрого и качественного рендеринга.

Отметим также, что цены на многие программные и аппаратные продукты для Amiga существенно ниже, чем на сравнимые по возможностям продукты с других платформ. Инструментальные средства для компьютеров можно условно разделить на три категории. Во-первых, это "consumer", "домашние" инструменты, которые доступны любому (зарубежному) пользователю — их цена измеряется сотнями или даже десятками долларов. Во-вторых, более сложные, профессиональ-

ные продукты для сферы бизнеса, "корпоративных" пользователей ценой в несколько тысяч долларов. Наконец, высокопрофессиональные средства, предназначенные для специализированных производств, для выпуска тиражных продуктов, средства, использование которых может приносить большие доходы,— здесь цены порядка десятков тысяч. Amiga росла "снизу", с рынка домашних компьютеров, и потому большинство продуктов для нее имеет цены из первого диапазона: постепенно возможности и самого компьютера, и новых версий инструментальных средств достигли уровня, сравнимого с возможностями профессиональных продуктов с других платформ — но порядок цен остался прежним. Сейчас, с фактическим выходом старших моделей Amiga на рынок профессиональных станций начинают появляться и системы, специально разработанные для этого рынка, с более высокими ценами.

Еще один фактор, обеспечивший успех Amiga на видеофронте — это уникальная по возможностям и цене плата VideoToaster фирмы NewTek. Она первоначально задумывалась как устройство для производства цифровых видеоэффектов (Digital Video Effects, DVE), однако в процессе разработки фирме удалось значительно расширить круг применений платы. История создания Toaster окружена романтичной легендой (см., например, Amiga World, July, 1991) о том, как часть персонала NewTek, "TeamToaster" (22 человека, в том числе 4 инженера, 10 программистов, 3 "программных дизайнера" и 5 разработчиков документации) в начале 1987 года удалилась от мира в секретное место, названное Alkatraz, где не было понятия "присутственное время", не было телефонных звонков и других отвлекающих моментов. Работая там в охотку по 70 часов в неделю, они "выжимали" из блиттера, коппера и другого амиговского хардвера невозможное, заставляя их работать в нештатных режимах, чтобы эмулировать дорогостоящее оборудование и снизить таким образом общую стоимость; параллельно создавались плата, программы, документация. Результат превзошел все ожидания. Были разработаны 4 СБИС-чипа, написано 350 тыс. ассемблерных команд — и в октябре 1990 года готовый продукт поступил в продажу.

Основа Toaster'a — это видеосвитчер, позволяющий плавно переходить с одного видеоисточника на другой, осуществляя при этом оцифровку кадров, их преобразование ("эффект") и вывод на видеоленту в реальном времени и со студийным качеством. Преобразования обеспечиваются DVE-генератором, в который заложено 128 "базовых" преобразований-эффектов! Путем комбинирования "базовых" эффектов с другими возможностями платы диапазон эффектов может быть еще расширен. Переходы могут осуществляться между четырьмя внешними источниками "живого" видео (камерами, магнитофонами), двумя кадровыми буферами (в которых могут содержаться "захваченные" кадры, сгенерированные компьютером изображения, тексты) и генератором цвета фона. Можно выбрать одну из встроенных скоростей перехода или задавать темп вручную, двигая мышью "Т-рукоятку" на экране.

Генератор символов (CG) может сгенерировать до 100 страниц текста с использованием 25 фонтов в 16 млн. цветов, с чрезвычайно четкими буквами, аппаратно обеспеченными эффектами вроде теней, плавного изменения цвета, постепенного "растворения" текущей страницы при переходе к следующей, скроллинга, бегущей строки и т.п. Frame Grabber позволяет сохранить текущий кадр в памяти (в 24-битном цвете). Toaster снабжен 2-Мбайт RAM, то есть включает два 24-битных кадровых буфера. Genlock позволяет накладывать графику Amiga поверх живого видео от любого источника, а также в комбинации с кадровыми буферами пускать анимацию поверх других амиговских картинок. Luminence Key накладывает живое видео поверх компьютерного фона или другого видеоисточника.

В комплект Toaster'a входят также три программы высокого качества — ChromaFX, ToasterPaint и LightWave 3D. ChromaFX — это цветовой процессор; он позволяет управлять яркостью, контрастностью, цветностью живого видео, превращать его в цветное негативное, в черно-белое, применять разнообразные цветовые фильтры и эффекты в реальном времени. LightWave — "полный" 3D-пакет, включающий Modeler, Renderer и Animator. По оценкам специалистов, он ни в чем не уступает, а во многом и превосходит другие 3D-пакеты на Amiga. ToasterPaint — 24-битная рисовальная программа с разрешением до 768x480 и рядом уникальных черт.

Доступ ко всем описанным аппаратным и программным возможностям обеспечивает основная управляющая программа. Благодаря комплексному дизайну Toaster'a все программные и аппаратные средства его динамически взаимосвязаны, из одной программы можно пользоваться возможностями другой, аппаратными хитростями, что делает работу удобнее и значительно повышает число доступных приемов и эффектов.

Самая ошарашивающая черта платы VideoToaster — ее цена: вместе со всеми программами — всего лишь 1595 долларов! Сравнимые по возможностям устройства DVE, по оценке Business Week Magazine, обойдутся в 60 000 долларов. "Голые" платы для других компьютеров, не обладающие и половиной возможностей Toaster'a, стоят в 2-3 раза больше него. VideoMagazine считает, что одна лишь программа ToasterPaint могла бы стоить больше, чем 1595 долларов (в самом деле, вспомните стоимость Lumena или 3D Studio на PC).

Единственным "недостатком" (в том лишь смысле, что нельзя сказать, что кроме Toaster'a ничего больше для видеостудии и не надо) является отсутствие в составе платы так называемого Time-Base Corrector (TBC), корректора временных искажений, устройства, необходимого для синхронизации видеоисточников. Однако тому есть объяснения: включение TBC удорожило бы плату, усложнило ее дизайн (и так Toaster конструктивно представляет собой "бутерброд" из двух плат, вставляемый в один слот), а многие пользователи вполне обходятся без TBC, работая с одним видеоисточником и осуществляя переходы с изображениями, хранящимися в кадровых буферах. При этом на рынке

существует большой выбор Toaster-совместимых TBC, от простейших одноканальных композитных устройств (ценой 850 долл.) до более дорогих (несколько тысяч долларов) моделей, имеющих два канала, умеющих работать не только с композитным, но и с S-VHS-сигналом и включающих также возможности транскодера и кадрового синхронизатора.

Сейчас выпущена вторая версия Toaster'a (2495 долл.) с такими, в частности, новыми эффектами, как "Fire, Liquid, Broken Glass".

К сожалению, давно обещанная PAL-версия Toaster'a так и не вышла. Возможно, аппаратные ухищрения разработчиков оказались настолько завязанными на параметры стандарта NTSC, что просто повторить их для PAL до сих пор не удается; возможно — и скорее всего — сыграли роль какие-то рыночные соображения (вспоминается сакраментальный вопрос: "Нэ могут или нэ хотят?"). Тем не менее пока европейским пользователям остается, с завистью (и все еще с надеждой) поглядывая за океан, строить свои системы на базе других 24-битных видеоплат. Диапазон таких плат — от дешевых HAM-E Plus (Blackbelt, 430 долл.) и DCTV (Digital Creations, 495 долл.) до видеоплат профессионального уровня: Harlequin (ASC, 1400-2000 фунтов стерлингов), Dual 32 (Progressive Peripherals), Impact Vision 24, или IV-24 (Great Valley Product, GVP, 2199 долл.), OpalVision (Centaur Development).

32-битный Harlequin в максимальной конфигурации включает два программно переключаемых кадровых буфера с разрешением до 910x576 (PAL) и два 8-битных альфа-канала. Альфа-канал может быть использован как 256-уровневый программноуправляемый линейный ключ для совмещения выхода Harlequin с внешним видеоисточником (от 0 — только Harlequin до 255 — только видео, с промежуточными. "полупрозрачными" состояниями). Имеется целый набор дополнительных устройств для Harlequin: покадровый контроллер VTR, фильм-рекордер, генлок, видеопринтер, frame-grabber, цифровой CCIR-601 видеоинтерфейс.

IV-24 может работать с одним 24-битным или двумя 12-битными изображениями разрешением до 768x576. Он включает средства захвата кадров, возможность воспроизведения "PIP"-видео ("picture-in-a-picture", "картинка в картинке"). К достоинствам платы относятся очень стабильное, "flicker-free" изображение, возможность ввода композитного или RGB-сигнала и вывода его как на multisync, так и на стандартный монитор, с одновременным наложением компьютерной графики; раздельные аналоговый композитный и цифровой RGB-генлок. Поддерживается вывод композитного, RGB и S-VHS форматов; в состав IV-24 входит VIU (Video Interface Unit) — видеоинтерфейсное устройство, обеспечивающее ввод и вывод видео в разных стандартах. IV-24 поддерживает 24-битный цвет на PAL-, NTSC- и VGA- (31 кГц) мониторах. С платой поставляются программные пакеты: Desktop Darkoom — программа обработки введенных изображений, аналог Photoshop; MyLAD (My Live Action Director) — видеосвитчер с 50 встроенными переходами; Macropaint-IV24 — 24-битная программа рисования и, кроме того, адаптированные к IV-24 версии пакетов Caligary-IV24 и Scala Titling-IV24. Как можно заметить, GVP явно пытается сформировать подобную Toaster "all-in-one" видеосистему.

Фирма Centaur тоже намеревается составить конкуренцию Toaster и построить полнофункциональную видеографическую систему OpalVision. Она будет иметь модульную структуру, и к уже выпущенной OpalVision Main Board (999 долл.) будут подключаться дочерние модули FrameGrabber/Genlock, Quad-input Production Switcher, Scan-Rate Converter (de-interlacer) и OpalVision Roaster Chip (прямой намек на намерение конкурировать с Toaster!).

Немного о программных продуктах.

Art Department Professional, ADPro (240 долл.) фирмы ASDG — один из лучших пакетов работы с изображениями и не только среди амиговских продуктов. Его модульная структура позволяет поддерживать множество форматов изображения, разные разрешения, разное число битовых плоскостей, преобразования между этими форматами, большой набор программ преобразования (операторов), различные программы загрузки и сохранения, в том числе в PostScript-формате, цветоделение, средства редактирования палитр и многое другое. К сильным сторонам пакета относятся богатые возможности композиции изображений. ADPro включает также мощные средства анимации, управляемые ARexx, редактор кадров FRED, другие инструменты разработки мультимедиа-презентаций и видеопрограмм. В настоящее время группа специалистов ASDG ведет работу над новым проектом, принципиально новой средой разработчика мультимедиа. По словам П.Киволовица (P.Kivolovitz), руководителя ASDG, создатели пакета мыслят в терминах "Время, Пространство, Вселенная"; по ряду намеков можно предположить, что принципы виртуальной реальности играют в проекте не последнюю роль.

Broadcast Titler 2 (InnoVision, 390 долл.) может в какой-то степени заменить пользователям PAL Character Generator Toaster'a.

Среди программных продуктов для Amiga отметим также 24-битный TVPaint, трехмерные пакеты Caligary Broadcast, Real 3D, Imagine, системы построения презентаций Scala и ShowMaker. Появились уже и программные JPEG-компрессоры, несколько программ двумерного морфинга; существует большое количество различных аниматоров, авторских систем, графических редакторов, музыкальных программ, всяческих недорогих аппаратных довесков, еще больше расширяющих способности компьютера. Читайте амиговские журналы!

Итак, Amiga, начав с рынка бытовых компьютеров, благодаря своей архитектуре и усилиям третьих фирм стала отличным видеоинтерфейсным компьютером для видеостудии, титровальной машиной, машиной видеоэффектов. Сейчас, с увеличением вычислительной мощности новых моделей и появлением профессионального уровня 3D-программ, Amiga становится пол-

нофункциональной рабочей видеостанцией, на которой можно выполнить весь цикл производства ролика, включая и качественный рендеринг. В любом случае, даже если основными компьютерами студии являются Silicon Graphics, Mac или PC, использование в студии компьютера Amiga позволит более оптимальным образом организовать технологический процесс.

Основным проводником линии Amiga в стране является петербургское А/О Creat ("Крейт"), а также его "креатуры" — Creat Graphics, АСКОД, АктИнвест. "Крейт" может также поставить различные комплекты Amiga, от A500 до A3000, в том числе полные интегрированные программно-аппаратные комплекты — "solutions", предназначенные для производства видеоанимации и мультимедиа-презентаций. Начинает поставлять видеостудии на базе Amiga и "Стиплер".

...плюс силиконизация!

"High End" рассматриваемого рынка — компьютеры Silicon Graphics, ценой в десятки и сотни тысяч долларов, специально разработанные для графических и анимационных применений. По программным и аппаратным возможностям они сегодня бесспорно лидируют. Об истории фирмы Silicon Graphics, Inc. (SGI), ее философии и дальнейших планах лучше всего прочесть в лекции основателя и председателя компании Джеймса Кларка (James Clark), в материалах конференции Графикон'91.

Весной этого года Silicon Graphics выпустила семейство компьютеров, которые представляют, по мнению фирмы, принципиально новый класс вычислительной техники — RISC PC (хотя можно вспомнить и Acorn Archimedes...). Станции семейства IRIS Indigo построены на базе RISC-процессоров MIPS R3000A (32-разрядный, 33 МГц) с FPU R3010A либо R4000SC (64-разрядный, 50 МГц) и образуют как бы две параллельные линии, каждая состоит из четырех моделей — просто Indigo, XS, XS24 и Elan. Производительность собственно компьютеров составляет 30 Mips/4.2 Mflops для R3000 и 85 Mips/16 Mflops для R4000, они имеют высокоскоростную шину (133 Мбайт/с у R3000, 266 — у R4000), 10 МГц SCSI-II интерфейс, скоростной интерфейс Ethernet, средства работы с качественным звуком (управляемые сигнальным процессором Motorola 56001) и много других замечательных свойств. Отмечается, что даже младшая, базовая модель Indigo R3000 во много раз быстрее не только 33-, но и 50-мегагерцевых 486 компьютеров.

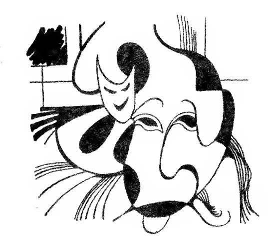
IRIS Indigo XS оснащена Geometry Engine — заказной СБИС, вдвое повышающей производительность 3D-вычислений. В XS24, кроме того, добавлены возможность работы с 24-битным цветом, альфа-канал и сглаживание. Elan также работает с 24-битным изображением, но оснащен четырьмя параллельными Геометрическими Машинами, что в 4 раза повышает скорость геометрических вычислений по сравнению с XS и XS24.

Конструкция станций семейства позволяет легко наращивать их мощность — вы можете превратить Indigo в Indigo XS, XS в XS24 и т.д. Однако переход с линии 3000 на 4000 гораздо сложнее — помимо самого процессора придется заменять и ряд других элементов, в частности блоки памяти, чтобы обеспечить работу с повышенной тактовой частотой. Дополнительная видеокарта IRIS IndigoVideo позволяет осуществлять ввод видео с трех независимых источников и вывод видео на экран в окна, которые могут динамически менять размер и расположение, а также имеет средства захвата кадров, генлок, видеовывод в разных форматах. К сожалению, у компьютеров Indigo всего два слота расширения, поэтому при работе с графической (Z-buffer) и видеокартой вы уже не можете, например, подключить дополнительную сетевую плату.

Станции IRIS Crimson построены на R4000 с удвоенной внутренней частотой — 100 МГц. Они способны обеспечивать плавное отображение сложных геометрических объектов в реальном времени.

Станции SGI поставляются с ОС IRIX, основанной на UNIX System-V с 4.3 BSD, POSIX 1003.I, X11 Window System, OSF/Motif, Display PostScript и многими другими стандартами "высшего уровня" мира UNIX. В состав IRIX входят также и собственные разработки SGI: знаменитая IRIS Graphic Library (IRIS GL), среда разработки мультимедиа-презентаций IRIS Showcase, средство построения приложений IRIS Explorer, набор утилит обработки изображений ImageVision Tools, многодорожечный звуковой редактор Audio Tools и др.

Аппаратные и программные средства Indigo разрабатывались с упором на максимальную производительность графических операций. Однако для быстрого старта в области трехмерной анимации профессионального уровня необходимы, помимо упомянутых базовых инструментальных средств, специализированные прикладные пакеты более высокого уровня. Основными разработчиками таких пакетов для станций Silicon Graphics являются фирмы Thomson Digital Images



(TDI), Wavefront Technologies, Alias Research, SoftImage, GIG. К сожалению, рамки обзора не позволяют подробно остановиться на этих высочайшего уровня инструментах, позволяющих конструировать и отображать синтетические миры с такой достоверностью, что, например, почти невозможно отличить синтетические образы, сгенерированные программами TDI, от фотографий реальных пейзажей, зданий, интерьеров. Наверное, имело бы смысл вернуться в будущем к станциям SGI и попросить кого-либо из специалистов, приобретших опыт работы с ней, рассказать более подробно как об архитектуре самих станций, так и о возможностях нрограмм TDI и Wavefront.

Заслуживала бы отдельного рассмотрения и рыночная политика самой SGI и упомянутых софтверных фирм. Ситуация здесь в корне отличается от рынка персональных компьютеров. Количество установок не так велико, степень защиты от копирования фактически 100%, поэтому рынок, сферы влияния жестко контролируются. Стоимость пакетов Wavefront и TDI сравнима со стоимостью самой станции (и даже порой скоррелирована с ней — к примеру, стоимость софтвера фирмы Wavefront тем выше, чем старше модель станции, для которой она покупается).

Silicon Graphics привезла несколько своих станций на Графикон'92 (где они оказались гвоздем выставочной программы) и вместе с представителями Wavefront и SoftImage провела сеанс "ловли на живца", подробно опросив и проанкетировав множество потенциальных заказчиков, слетевшихся на демонстрацию, как бабочки на свет; после этого "наживка" — все 6 машин ELAN 3000 — была упакована и отправлена обратно в Штаты.

Вопрос с ценой станций Indigo не вполне ясен. Согласно данным прессы (например, Infoworld, May 11, 1992), американские "retail"-цены четырех моделей R3000 соответственно 7995, 13 000, 15 000 и 27 000 долларов, причем это цены не "голых" системных блоков. Например, за 7995 долларов вы получаете Indigo с 8-Мбайт RAM, 16-дюймовый монитор Mitsubishi, 1024x768 цветную 8-битную графику, клавиатуру, мышь; в 27 000 долл. для ELAN укладываются также 16-Мбайт RAM, 432-Мбайт SCSI-винчестер, 150-Мбайт стример и 19-дюймовый 1280x1024 цветной монитор. Но цены, по которым это семейство предлагалось на Графиконе, были заметно выше. Если же учесть еще и стоимость прикладных программ, видеокарты, общая стоимость установок на основе старших Indigo приблизится к 100-тысячному рубежу. Будем все же надеяться, что фирме удастся быстро и благоприятно для нас проанализировать результаты собеседований, уладить вопросы с КОКОМ, установить гуманный уровень цен — и уже серьезно выйти на наш рынок.

Тем временем российский рынок вполне созрел для этой техники и даже для соответствующего ей уровня цен. Уже весной прошлого года отдельные студии посматривали в сторону SGI, некоторые специально посетили штаб-квартиру фирмы в США. Сейчас же почти все лидеры рынка, быстро проскочив IBM'овский или Amig'овский период, всерьез намереваются приобретать станции SGI, для них это вопрос решенный. При этом многим уже мало R3000, они сетуют на КОКОМ, ограничивающий пока продажу ELAN 4000 и Crimson. Более того, есть новички с серьезными намерениями (и, очевидно, серьезными финансовыми возможностями), которые собираются стартовать в видеорекламе с нуля — и сразу на старших моделях Silicon. Если процесс "силиконизации" страны будет идти в таком темпе, образовавшиеся производственные мощности в принципе смогут через год—полтора обеспечить заполнение 95% эфирного времени по всем ТВ-программам рекламными роликами нездешнего качества. Однако поскольку оставшихся 5% будет явно недостаточно для демонстрации многочисленных "Богатых" и "Санта-Барбар", у изготавливающих ролики студий могут возникнуть проблемы с загрузкой и окупаемостью станций — и тогда, есть надежда, доступ к завезенным в период "рекламного бума" станциям SGI получат (тем или иным образом) ученые, дизайнеры, архитекторы, для которых сегодня и AT/286 часто большая удача, а уж Mac — явный предмет роскоши.

К минусам станций Indigo обозреватели относят их узкую специализацию: диапазон использования пока сильно ограничен малым выбором софтвера, не относящегося к графическим применениям, поэтому если у обладателя Indigo вдруг почему-либо не заладились дела на графическом фронте, найти с ходу другую нишу не так просто. Для Sun и IBM R6000, о которых мы поговорим в дальнейшем, ситуация совершенно иная. Впрочем, SGI объявила о ведущейся разработке эмулятора DOS, с помощью которого можно будет запускать программы, написанные для 386-х и 486-х компьютеров.

Промежуточным вариантом для не очень богатых студий могла бы стать плата IrisVision для IBM PC, которая позволяет на 486-м компьютере работать как с привычными 3D-Studio и AutoCAD, так и с программами IRIS Graphics Library. Плата построена на основе Геометрической Машины и еще 6 заказных чипов, разработанных SGI. Сейчас, однако, Silicon Graphics продала права на IrisVision другой фирме, и дальнейшая судьба платы неясна.

Полной ясности с поставками Silicon Graphics пока нет, однако фирмы "Элогар", Render-Club, "Крейт", "Тивионика" уже заявили о себе как о возможных поставщиках станций Silicon (в комплекте с софтвером Wavefront, SoftImage, TDI). Возможно также (по непроверенным данным) приобретение станций SGI в составе интегрированной видеоанимационной системы у поставщиков профессиональной видеотехники.

Последние новости в этой области: в ноябре прошла презентация продуктов TDI, проведенная представителем фирмы и Render-Club; после презентации Elan-4000 временно остался в "Клубе", где его можно было посмотреть в более спокойной обстановке. Вопрос с КОКОМ Silicon Graphics обещает утрясти к январю; тут-то и начнется "марш слонопотамов" — шествие Indigo на наш рынок...

Конверсия и анимация

Еще одно, стоящее особняком, направление на рынке компьютерной анимации — "побочный продукт" ВПК — студии, созданные вокруг сверхмощных (по тем временам) спецкомпьютеров, использовавшихся для моделирования и тренинга в авиации и космонавтике. Стоимость таких машин оценивается в сотни тысяч долларов. Они позволяют выполнять обсчет тысяч полигонов, теней, освещения за 1/25 секунды, обеспечивая таким образом анимацию в реальном времени. Это делает работу художника значительно более интересной и творческой — появляется возможность управлять объектами в интерактивном режиме. Вокруг этих компьютеров — сложившиеся коллективы программистов, решавших сложные задачи управления, моделирования и отображения в реальном времени, наработанные методы и библиотеки программ. Весь этот накопленный багаж может быть с успехом применен и в производстве анимационных фильмов. Подтверждение тому — приз, завоеванный фильмом "Shadow" ("Тень") студии "Альбатрос" (Новосибирск) на фестивале компьютерного искусства Ars Electronica в Австрии в 1991 году.



Здесь стоило бы сказать и о системах на базе транспьютеров, также позволяющих просчитывать в реальном времени чрезвычайно сложные сцены. Однако, к сожалению, информации об использовании таких систем конкретными командами и студиями нет, хотя можно предположить, что именно транспьютеры используются для решения подобных задач в современных военных и космических приложениях.

Сейчас команды специалистов из Новосибирска ("Альбатрос" и SoftLab) всерьез заинтересовались вопросами виртуальной реальности. Имеющаяся в их распоряжении технология, пожалуй, наиболее соответствует этой тематике. Рассматривается вопрос о создании секции виртуальной реальности при недавно образованной Московской группе ACM SIGCHI.

С.Новосельцев
Тел.:938-66-10
E-mail:next@ipian15.ipian.msk.su

(Продолжение следует)



Колонка редактора

Мне бы хотелось поговорить о том, что нас ожидает в этом году. Прошедший год ознаменовался крупными перемещениями среди основных фирм — разработчиков программных продуктов. Слияния, поглощения, объединения продолжались в течение всего года, но основные результаты этих действий будут видны только в этом и следующих годах. На мой взгляд, интересным ожидается альянс Clarion + JPI и Borland + AshtonTate.

Продолжается наступление многооконных графических сред; в числе серьезных претендентов на первенство в этой области — DesQview/X фирмы Quarterdesk. Сетевое расширение Miscrosoft Windows — Windows for Workgroups стало еще одним шагом на пути к Windows NT — настоящей 32-разрядной сетевой операционной системе. Отмечу, что основной проблемой, сдерживающей рост программного обеспечения, является платформа — операционная система DOS, по-прежнему работающая в реальном режиме процессора. От этого все беды.

В этом году мы планируем опубликовать ряд обзоров средств для программистов. В ближайших номерах вы познакомитесь с библиотекой ObjectProfessional фирмы TurboPower Software и графической библиотекой TEGL. Не останутся без внимания и другие полезные для разработчиков программные продукты. В наших планах — статьи по программированию в среде Windows, расширению функциональности Turbo Vision, языку Actor, пакету ToolBook и многое, многое другое.

Будет возобновлена рубрика "На книжной полке". В ближайшее время мы рассмотрим в ней ряд книг, посвященных использованию среды Windows и программированию в ней.

А. Федоров

Новости

Если вы едете на машине и хотите при этом работать на вашем компьютере-блокноте, американская фирма Wen Technology, кажется, сможет помочь вам.

На выставке Comdex Fall она продемонстрировала управляемый голосом компьютер-блокнот иа 486-м процессоре — снабженный микрофоном, наушниками и специальным программным обеспечением распознавания и синтеза голоса. Президент и основатель компании заявил, что это устройство можно применять где угодно, даже в шумном автомобиле.

После недолгого периода обучения машины вашему голосу становится возможным отдавать команды DOS'у или Windows "вслух". Например, можно обучить программу реагировать на слова "послать электронную почту" или "включить факс". Синтетический голос может предоставлять пользователю специальные меню.

Новый блокнот сможет работать и как обычный компьютер... Весит он 6.2 фунта (2.8 кг), имеет 2 Мбайта ОЗУ, 60-Мбайтный винчестер, 85-клавишную клавиатуру, 3.5-дюймовый дисковод, съемную батарею, 8.5-дюймовый жидкокристаллический экран, обеспечивающий 256 цветов при разрешении 320x200 точек, и 16 цветов при разрешении 640x480. Он будет продаваться оптовыми партиями по цене 4500 долларов в минимальной конфигурации.

Newsbytes News Network,
November 24, 1992

Всеми ожидается, что в начале февраля на выставке MacWorld в Японии Apple выпустит на рынок свой первый цветной компьютер-блокнот PowerBook и еще шесть новых машин.

По слухам, PowerBook 165c будет цветной версией PowerBook 160, поддерживающей пассивный жидкокристаллический дисплей с разрешением 640x400 точек. Более быстрая и более дорогая версия с процессором на 33 МГц ожидается позже в этом году.

Основная конфигурация 165c — 4 Мбайта ОЗУ и винчестер 80 Мбайт — и всего-то за 3600 долларов.

Одновременно с цветным PowerBook ожидается выпуск нового Mac Classic, который предположительио будет продаваться за 1499 долларов. Также ожидается цветной Mac LCIII 25 МГц, который должен снизить цену на цветной Макинтош до 1500 долларов, а также три новых системы на процессоре 68040.

Apple хочет начать новый год с большого выстрела новыми продуктами по публике. В январе на MacWorld в Сан-Франциско фирма продемонстрировала два новых лазерных принтера, 24-битный цветной сканер и программы обработки цветных изображений.

Newsbytes News Network,
January 8, 1993

Фирма Zeos решила практически полностью обновить ассортимент своей компьютерной продукции. По заявлению президента фирмы Грега Херрика, они выпускают на рынок аж 20 новых машин на процессоре 80486, которые на 18 процентов дешевле ранее продававшихся. Все машины будут иметь шину VESA, новые видеорежимы и высокоскоростной дисковый интерфейс в стандарте IDE.

Новые модели делаются на базе 486SX, DX и DX2 процессоров фирмы Intel, работающих на частотах 25, 33, 50 и 66 МГц. Каждый процессор обрамляется одной из четырех стандартных комбинаций периферии — от 2 Мбайт ОЗУ и 85-Мбайтного винчестера за 1395 долларов, до 16 Мбайт ОЗУ и винчестера 340 Мбайт за 2995 долларов.

По заявлению фирмы, цены уменьшены примерно на 350 долларов на все машины.

За дополнительные 150 долларов фирма начала поставлять видеоускоритель для Windows, который обещает вчетверо увеличить скорость оконной графики.

Из других новостей фирма сообщила о переезде из пяти зданий в одно большое, а также о начале массовой почтовой рассылки своей рекламы примерно 350 тысячам бывших и потенциальных будущих клиентов.

Newsbytes News Network,
January 6, 1993

Всюду воруют, или Аресты в Силиконовой долине

Компьютеры и программы являются крупным источником доходов для большинства предприятий Силикоиовой долины в Калифорнии. Эти же нродукты дают хорошие доходы и преступным элементам. Местная и федеральная полиция вместе с ФБР провели недавно секретную операцию, в результате которой было арестовано 30 человек.

Результатом продолжавшейся четыре с половиной месяца операции "Серый чип" было обнаружение украденного ранее в разных местах компьютерного оборудования на сумму 3.6 миллиона долларов. По сообщению, примерно 2.1 миллиона было возвращено владельцам в виде денег за уже проданные краденые машины, и еще полтора миллиона — своровапными, но еще не проданными компьютерами.

Агентство UPI цитирует сержанта Марка Керби, который заявил, что "торговля ворованными компьютерами и их частями является одной из основных проблем в регионе. Мы задели только верхушку айсберга. Но это должно послужить предупреждением остальным жуликам".

Преступников арестовывали в разных местах — от автостоянок до ресторанов — причем практически всех взяли с поличным.

В декабре 1991 года агентство Newsbytes сообщало о том, что у фирмы Acer при перевозке со склада на склад было украдено компьютеров-блокнотов на сумму в четверть миллиона долларов.

В феврале 1992 года примерно 20 или 30 компьютерных магазинов в Калифорнии крупно погорели следующим образом: некто приходил в праздник с банковским чеком и просил принять его в качестве оплаты за мегабайтные микросхемы памяти. Как правильно выглядевшую бумажку, чек принимали, товар выдавали, а на следующий день обнаруживали, что их жестоко обманули и денег за него не получишь... В каждом магазине было таким образом украдено чипов на 5-10 тысяч долларов...

Newsbytes News Network,
January 13, 1993

Dell тоже снижает цены

На 18 процентов. И собирается выпустить целое лукошко новых 486-х машин. Все они будут иметь возможность простого перехода на работу с процессором Pentium фирмы Intel.

Новая линия Dell Dimension состоит из пяти настольных систем, трех напольных серверов и двух блокнотов. Все системы поставляются с процессором 386 или 486, монитором, видеоадаптером SuperVGA с ускорителем и разрешением 1024x768, мышкой, DOS'ом и Windows 3.1.

Dell заявляет, что ускоряет процесс выполнения заявок, снижает количество промежуточных звеньев, усиливает проверки качества и т.д. Говорят, что теперь все владельцы машин Dell всегда смогут спрашивать совета насчет решения проблем с их компьютерами по бесплатному телефону фирмы. Пока не умрут либо они, либо фирма...

А еще Dell выпустила Multimedia PC. Машина имеет 4 Мбайта ОЗУ, жесткий диск емкостью 80-170 Мбайт, флоповод, SuperVGA, карту SoundBlaster и стереодинамики, привод CD-ROM, микрофон и модем.

Фирма подстраивает свои изделия под три сегмента рынка — машины для школьников (KidStation system), машины для студентов (StudentStation system) и OfficeStation — для совсем взрослых. Каждый вариант будет иметь свой набор заранее установленного программного обеспечения.

Например, на StudentStation (386SX/ 120 Мбайт винчестер/ 512 Кбайт видеоОЗУ/ CD-ROM/ 1999 долларов) будет устанавливаться DOS 5.0, Windows 3.1, Windows Draw, Multimedia Microsoft Works, Compton's Multimedia Encyclopedia, Sherlock Holmes Consulting Detective, Where in the World is Carmen San Diego? Deluxe, Just Grandma and Me, Microsoft Bookshelf for Windows, и Wordstar Writing Tools for Windows.

Для взрослых предлагается набор из игр, энциклопедии и программ для подсчета денег.

По заявлению Dell, исследования независимой исследовательской фирмы Gather Group показали, что к 1996 году на каждой четвертой поставляемой машине будут гонять программы Multimedia.

Newsbytes News Network, December 1, 1992

Подешевели блокноты Compaq

Процентов на 12-16, в зависимости от модели. Машины с цветным экраном подешевели тоже.

Цветные машины теперь можно купить примерно за 2000 долларов, а простой блокнот на базе 386SL можно приобрести за 1250. Compaq заявляет, что теперь их машины на 10-25% дешевле, чем у AST, Dell или Toshiba (до аналогичного заявления конкурентов).

Кажется, фирме удалось справиться с большим количеством задержанных и неудовлетворенных заказов. Компьютеры серии Contura изготавливаются на заводах фирмы в Сингапуре и Шотландии. Говорят, что конвейеры работают 24 часа в сутки.

Это уже второе снижение цен на блокноты Compaq Contura.

Newsbytes News Network, January 20, 1993

Киевское предприятие "Диалектика" выпустило в свет книгу Гради Буча "Объектно-ориентированное программирование с примерами применения". Книга, являющаяся первым изданием, посвященным объектно-ориентированной методологии создания программного обеспечения, состоит из трех частей. В первой части обсуждается концепция создания модуля реального объекта, приводится анализ развития языков программирования и рассматриваются элементы объектно-ориентированного подхода: объектно-ориентированный анализ, проектирование и программирование. Вторая часть посвящена подробному рассмотрению вопросов объектно-ориентированного проектирования. В третьей части книги приводятся примеры применения изложенных концепций на различных языках: SmallTalk, Object Pascal, C++ и Ada. Книга может быть полезна как профессиональным программистам, так и студентам, осваивающим технологию объектно-ориентированного программирования.

Научно-проектная фирма "Диалектика" занимается разработкой программных средств для программистов, изысканиями в области искусственного интеллекта, реализацией новых технологий программирования и выпуском литературы соответствующей тематики. Ею были выпущены справочники по библиотеке Turbo Vision для компиляторов Turbo C++ и Turbo Pascal, а также сборник статей, посвященных стандарту CUA/SAA фирмы IBM.

Адрес фирмы "Диалектика": Украина, г. Киев, ул. Пархоменко 14.

КомпьютерПресс, февраль 1993

Фирма CompuAdd выпустила два новых компьютера на базе процессоров Intel 486DX2, работающих на частоте 66 МГц. CompuAdd 466 использует шину ISA, а CompuAdd 466E — более производительную EISA. Обе модели доступны как в виде настольной машины, так и в корпусе "башня". Обе модели поддерживают до 64 Мбайт оперативной памяти, поставляются с самыми разными по емкости винчестерами и различным объемом ОЗУ. Предусмотрен кэш объемом до 64 Кбайт.

Базовые конфигурации обойдутся покупателю в 2640 или 2820 долларов для 466 и 466E соответственно.

InfoWorld Direct, November 1992

"Я верю, перья будут!"

В конце декабря в Москве открылся магазин фирмы **Rotring** — всемирно известного изготовителя перьев для графопостроителей. Сегодня Rotring предлагает перья для тысяч моделей плоттеров ста шести фирм. Среди них как самые известные — такие как Hewlett-Packard, IBM, Benson, Data General, Siemens, Quest, Roland — так и самые экзотические.

Как рассказал представитель Rotring в Москве Александр Савин, раньше перья Rotring поставлялись очень многими фирмами под их собственными названиями, но сейчас фирма стремится продавать их со своим фирменным знаком. В мире Rotring обслуживает около 70% этого сектора рынка, а среди всей продукции фирмы перья для графопостроителей составляют 80%.

Приятно, что в новом магазине работает грамотный персонал, способный квалифицированно ответить на самые разные вопросы. В случае необходимости поставки принадлежностей к редко встречающимся графопостроителям вам помогут правильно оформить заказ.

Разумеется, кроме самих перьев, есть все необходимые дополнения к ним — тушь, колпачки, адаптеры, даже специальные ластики для внесения изменений в чертежи на пленке. Кроме того, предлагается весь диапазон продукции Rotring, включающий в себя инструменты и материалы для черчения и рисования, предназначенные конструкторам, архитекторам, дизайнерам, художникам. Оплата возможна как в рублях, так и в СКВ, как по перечислению, так и за наличный расчет.

Магазин расположен на ВДНХ в павильоне "Зерно", телефон 282-38-46. Телефоны представительства Rotring в Москве 201-56-16, 291-15-28.

КомпьютерПресс, январь 1993

К.Чащин, А.Федоров, И.Вязаничев

АРМ

001 OK
Автоматизация работы отдела кадров предприятия.
Круг 14800р.*(1)

ТЕКСТ-ПРОЦЕССОРЫ

002 ETXT
Пакет для обработки текстовой и графической информации. Редактор текстов ET.EXE, графический редактор BWI, конвертор графических файлов CONV составляет комплексную систему для получения текстово-графических документов.
С.Дорошев 2190р.*

003 ETXT-LQ
Пакет для обработки текстовой и графической информации. В пакет включен текстовый редактор ET-LQ со значительно расширенными функциональными возможностями, который позволяет получить на матричных принтерах печать близкую по качеству к лазерной.
С.Дорошев 3860р.*

ИНСТРУМЕНТЫ

004 MENU FOR PROGRAMMERS
Организация диалога типа МЕНЮ из прикладных программ (MS Fortran, MS C, Turbo C, Turbo C++, Turbo Pascal).
А.Владимиров 3800р.(1)

005 KITE INTERFACE MAKER v1.0
Библиотека текстового интерфейса для Turbo C, Turbo C++.
А.Коршунов 2200р.(2)

006 PROKUS48 v2.05
Интегрированная инструментальная система для проектирования и отладки устройств на ОМЭВМ ряда 1816ВЕ35 39 48 49.
ТОО "МТИ" 11050р*.(1)

007 XSKit
Библиотека функций Си и Паскаля для чтения данных из незащищенных паролем таблиц Paradox Ver 3.0, 3.5.
Мозговой В.И. -.

008 Text_HIStorian
Программа архиватор версий программных продуктов для одиночной и групповой работы. Восстановление версий, сохраняемых инкрементно.
Бахов В. 1500р.(1)

ПРИКЛАДНЫЕ ПРОГРАММЫ

009 ISS1.0
Пакет "Интерпретированные структурные схемы" позволяет обрабатывать деревья и последовательности опирающихся друг на друга деревьев-размеченные частичные порядки.
Приходько А.Н. -.

ЗАЩИТА

010 PROTECTION SYSTEM
Защита .EXE-файлов от несанкционированного копирования. 3 типа защиты.
HardSoft 3400р.(1)

011 KEYDISK
Создание ключевой дискеты и привязка к ней исполняемых программ.
HardSoft 1600р.(1)

012 HARDLOCK
Индивидуальный и многопользовательские пароли на жесткий диск.
HardSoft 1400р.(1)

УТИЛИТЫ

013 HS DRIVER
Система поддержки национальных алфавитов. Драйвер EGA, VGA, KBD. Редактор шрифтов, планировщик клавиатуры.
А.Аникин 2600р.(1)

ГРАФИКА

014 GRAPHIC INTERFACE
Экспорт, импорт и программная поддержка работы с изображениями.
HardSoft 5300р.*

015 MS-GRAFOR 4.0
Пакет графических подпрограмм на MS FORTRAN'е 5.0 (вывод на экран графиков функций 1-го и 2-х аргументов, 3 мерных объектов, текстовой информации). VGA, EGA, CGA, 16 цветов; 6 русифицированных и масштабируемых шрифтов.
Гринин С.В. - *

ОБУЧАЮЩИЕ ПРОГРАММЫ

016 PDD
Программа обучения и проверки знаний правил дорожного движения. EGA 640x350,16цв.
Баев А.В. 10000р.*(6)

017 TEMA
Автоматизированное обучение и контроль знаний более 40 различных тем.
Цимфер А.А. 3900р.(1)

ИГРЫ

018 STRIP CROSS PLATE
Эротическая головоломка.
А.А.А. 500р.(1)

019 Centaur
Уникальная шахматная программа, 5-кратный чемпион Москвы, оригинальный алгоритм, широкий выбор вспомогательных функций, рейтинг - 2200.
Вихреев В.В. 7000р.(1)

БАЗЫ ДАННЫХ

020 LASABASE
Программа-каталог для обработки научной и библиографической информации.
Влад И. Мир -.(1)

HARDWARE

021 StarCom
Адаптер последовательных каналов для IBM PC AT/XT 8ИРПС или 6ИРПС + 2RS232. Скорость до 19200 б/сек. Работа по прерываниям. Дискета с примерами.
АО МИКА СНТ -.(1)

Уважаемые читатели!

Предлагаем вашему вниманию очередной список наиболее популярных программ, представленных в "HS Listing".

HS BOARD

1. Strip Cross Plate
2. Protection System
3. PDD
4. Menu for Programmers
5. GRAPHIC INTERFACE
6. KANDIDAT
7. TVESA
8. POISK-18
9. MEMORY METER
10. CL

По независящим от нас причинам временно прекращен прием заявок на размещение информации о программных продуктах. О возобновлении приема мы сообщим дополнительно.

Обозначения: '-' — "HS Listing" продуктом не располагает; '*' — наличие демо-версии; '()' — количество дискет, необходимых для размещения продукта (стоимость каждой дискеты — 60 руб).

Наш адрес: 109180 Москва, а/я 34 "HS Listing"

Лучшие программные продукты от фирмы Computer Associates

CA-Clipper/Compiler Kit — компилятор dBASE IV

CA-Clipper/Compiler Kit для dBASE IV создает автономные выполняемые .EXE-файлы, которые можно свободно распространять, из Ваших dBASE IV программ.

CA-Clipper/Compiler Kit автоматизирует процесс компиляции dBASE IV-приложений, используя мощный компилятор пакета CA-Clipper. Никакие предварительные знания о CA-Clipper не нужны. Ничего не нужно менять в исходных текстах dBASE IV-программ (.PRG). Ваши приложения приобретут дополнительную гибкость и удобство при распространении и использовании.

CA-SuperProject — самые эффективные алгоритмы планирования при управлении проектами

Если Вы менеджер с современным стилем работы и хотите добиться выполнения стоящей перед Вами задачи, обеспечив высокое качество и уложившись в сроки и бюджет, то SuperProject поможет Вам в этом. SuperProject позволит принять наиболее эффективное решение, определив оптимальную последовательность работ и распределение ресурсов. Даже если ресурсы, имеющиеся в Вашем распоряжении, ограничены.

Для работы с SuperProject Вам не потребуется никаких специальных знаний.

CA-Clipper Tools II for DOS — лучшая библиотека функций для CA-Clipper 5.0

CA-Clipper Tools II — инструментальная библиотека для CA-Clipper 5.0, созданная для поддержки профессиональных разработчиков, открывает программистам Clipper новые области, доступные прежде лишь тем, кто обладал достаточно глубокими знаниями о системе или имел большой опыт работы на языках Ассемблера или Си.

- Более 550 функций, разбитых на 17 категорий
- Справочные системы, примеры и утилиты
- Сотни проверенных решений

CA-SuperCalc — электронная таблица будущего на Ваших компьютерах сегодня

CA-SuperCalc — это профессиональный табличный процессор, разработанный с таким расчетом, чтобы даже новичок мог в полной мере использовать все возможности своего компьютера.

CA-SuperCalc сочетает простоту в использовании с уникальной мощью развитого трехмерного моделирования, связывания электронных таблиц, построения разнообразных диаграмм, высококачественной печати, реляционной базы данных и встроенных средств отладки, поддержки локальных сетей.

Приглашаем Вас на наш стенд №216 на выставке Comtek'93!

С 1993 года филиал фирмы Computer Associates в России издает специальный журнал Computer Associates News, посвященный продуктам фирмы. Подписка свободная.

Россия, 127018 Москва, 2-я Ямская, 15
Телефоны: (095) 289-44-77, 289-44-83
Факс: (095) 289-34-69

